51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2013年01月
总第022期

《2013年开发者之路》

应届生从微软到谷歌求职之路

面试程序员时的“唠叨”

目录

2013.01 Contents

编程排行 Billboard

专题报道 《程序员的道路》

技术热点 Techlogy hot

51CTO.com
技术成就梦想

出版方 :北京无忧创想信息技术有限公司

责任编辑 :彭凡 林师授 张伟 封面设计 :苍旭

联系方法 :zhangwei@51cto.com 010–68476606 (分机 8110)

出版日期 :2013 年 1 月 18 日

欢迎广大读者投稿

编者按

Objective-C 不但两次赢得了 TIOBE 编程语言奖项，并且在 2012 年编程语言中也获得了大部分市场的份额。Objective-C 能够获此佳绩，其重要的原因是目前手机应用开发蓬勃发展所带来的。

2013年1月编程语言排行榜：移动编程语言的巅峰

【51CTO 独家特稿】Objective-C 不但两次赢得了 TIOBE 编程语言奖项，并且在 2012 年编程语言中也获得了大部分市场的份额。Objective-C 能够获此佳绩，其重要的原因是目前手机应用开发蓬勃发展所带来的。

另外，在 2012 年有几个有趣的浮动，C++（+1.09，主要是由微软带动），Python（+0.96%），另一方面，C#（-2.57%，主要由于进入移动市场比较晚）和 Delphi（-0.65%）也失去了很大一块市场份额。

那么在 2013 年编程语言排行榜应有什么样的动作呢？首先，随着手机应用程序市场的继续增长占着主导的地位，预计 Java（Android）和 C++/C#（Windows Phone）也将会重获民心，而 Objective-C 也会继续增长。

其次，JavaScript 和 MATLAB 的趋势也很乐观。

JavaScript 几乎在任何程序中都发挥着越来越重要的作用。

而 MATLAB 事实上在过去的几年中已成为国际控制界的标准计算，MATLAB 的应用范围非常广，包括信号和图像处理、通讯、控制系统设计、测试和测量、财务建模和分析以及计算生物学等众多应用领域。

2013 年 1 月编程语言排行榜榜单

Position Jan 2013	Position Jan 2012	Delta in Position	Programming Language	Ratings Jan 2013	Delta Jan 2012	Status
1	2	↑	C	17.855%	+0.89%	A
2	1	↓	Java	17.417%	-0.05%	A
3	5	↑↑	Objective-C	10.283%	+3.37%	A
4	4	=	C++	9.140%	+1.09%	A
5	3	↓↓	C#	6.196%	-2.57%	A
6	6	=	PHP	5.546%	-0.16%	A
7	7	=	(Visual) Basic	4.749%	+0.23%	A
8	8	=	Python	4.173%	+0.96%	A
9	9	=	Perl	2.264%	-0.50%	A
10	10	=	JavaScript	1.976%	-0.34%	A
11	12	↑	Ruby	1.775%	+0.34%	A
12	24	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	1.043%	+0.56%	A
13	13	=	Lisp	0.953%	-0.16%	A
14	14	=	Pascal	0.932%	+0.14%	A
15	11	↓↓↓↓	Delphi/Object Pascal	0.919%	-0.65%	A
16	17	↑	Ada	0.651%	+0.02%	B
17	23	↑↑↑↑↑↑	MATLAB	0.641%	+0.13%	B
18	20	↑↑	Lua	0.633%	+0.07%	B
19	21	↑↑	Assembly	0.629%	+0.08%	B
20	72	↑↑↑↑↑↑↑↑↑↑	Bash	0.613%	+0.49%	B

MATLAB 介绍

MATLAB 是由美国 mathworks 公司发布的主要面对科学计算、可视化以及交互式程序设计的高科技计算环境。它将数值分析、矩阵计算、科学数据可视化以及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计以及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，并在很大程度上摆脱了传统非交互式程序设计语言（如 C、Fortran）的编辑模式，代表了当今国际科学计算软件的先进水平■原文未完

http://developer.51cto.com/art/201301/375900.htm

年关将近，很多人开始寻思的改如何找下家，对于人员的正常流动是很常见的，这里就不多说了，就跳槽的原因有以下几点感想。

一个就是不要因为人际关系跳槽，不要因为看不惯某个同事、某个领导跳槽，有人的地方就有江湖。就算换一个新的环境一样可能遇到你看着不爽的人，你可以把这当成一个挑战，然后去战胜它，试着“搞定”这个人。其实就是缓和你们的气氛，达到彼此的沟通，能够获得对方的支持。有的时候，并不是那么好操作和实现，你需要有很高的度量，但是当你做到了，难道不是一种成长吗？

与其在乎薪水，不如在乎自己创造的价值。

在自己干得好的时候跳，而不是干得不好的时候。听上去很傻，但却是这样。如果你在这家公司做得不好，凭什么换一家公司就能做得好了呢？如果你在这家公司做得非常好，下一家公司有什么理由怀疑你在这里就做不好了呢？

所以，跳槽前，至少给自己一个满意的答卷。而且毕竟相聚一场，如果你为上家公司做了比较大的贡献后离开，大家也很容易成为朋友。

你怎么知道这些人当中某位十年后不会成为自己的贵人呢？所以，即便离开，和大家保持一个良好的关系是非常重要的。

专题地址 :http://developer.51cto.com/exp/mszjh/index.html

51CTO 开发频道语

2013年开发者之路

当玛雅人晃点全世界后，我们还需要正面世界末日后的生活。既然我们选择要好好活下去，各位开发者在 2013 年该如何走呢？

深冬的丰台花园，池塘中的水已经被低温彻底冻住，小孩儿们正高兴的在上面练习滑冰。已经到来的 2013 年，我们能看到一丝丝寒意。那 2013 年，我们该有怎样的规划呢？

1. 给 WEB 前端工程师

WEB 前端工程师已经感受到越来越大的压力。不同浏览器的区别，让有些人会说，浏览器是他所知道的开发平台中最不稳定的一个。兼容性问题有时会让人崩溃。如果你是一个前开发工程师，理解浏览器内部工作原理会帮助你作出更好的决定。

前端开发工程师如何在 2013 年里提升自己

2013 年你应该更好的了解 HTML 5。尽管他还存在两条标准的分裂问题，还有不同平台上性能不足的问题。但你不得不承认，未来确实在 HTML 5 手上。越来越多的开发者不得不学习好 HTML 5，以应对未来的变化，特别是移动互联网的变化。我们需要关注很多解决现实世界问题的方案。你会了解媒体流，设备输入，现代 CSS 设计，媒体捕捉，文件 I/O 等等。

2. 给二十多岁的开发者

当一个人步入 30 岁的而立之年，你的大脑已经趋于稳定。倘若 2013 年你还处于 30 岁之前，那恭喜你，你还正处在开发者最变化多端的几年。

如果你希望学习更多开发知识和管理技能，这几年将是你积累最迅速的时期。因为再过几年你就很难或者根本不可能像现在这样学习知识。听到很多的创业者在公司做到一定规模后就没有能力（或不想）管理这个公司。这就是岁月不饶人的代价。

所以，当你在 2013 年的寒冬中不幸被裁员。说不定这是你最好进行知识积累的时刻，别放弃。

3. 移动互联网到底适不适合你

智能手机让大家看到了商机，似乎做个好应用，金疙瘩就会砸到开发者头上一样。很多开发者，特别是 30 到 40 岁年龄段的开发者，都会计划在 2013 年开始自己的创业之旅。不过，当老板就可以解脱了吗？

——2011 年 8 月，互联网著名人士刘兴亮宣称以“拿起一块名叫闪聚的板砖，狠狠地敲开了移动互联网的大门”，高调宣布其移动互联网创业项目“闪聚”，2011 年 11 月，iOS 公测版上线，其为一款 LBS 移动交友应用，通过共同感兴趣的约会帮助近距离陌生人结识相聚，主打陌生人交友。刘兴亮此前曾表示创业资金来源于一位老友，足够维持一年的运营开支，后被传融资约 500 万元人民币。

然而在一年过后，据内部人士透露，“闪聚”

2013 年开发者之路 II

早在数月前关闭，其在苹果 App Store 中国商店也已经下架，尚存于 Android 商店的 App 也被爆有闪退、无法注册等多处严重 BUG。据知情人士透露，2012 年 5 月底，闪聚发布 iPhone 1.2 版本后，便再无消息出处，不久后团队宣告解散，创业历程不足 1 年。闪聚官网 (shanju.com) 已经无法打开，联系电话显示为空号，官方微博最新一条消息停留在 2012 年 5 月 21 日。

这样的报道只是众多死掉的创业团队中的一个。很多创业团队犹如 D Day 中被困在奥哈马海滩的美军一样，成为融资难、变现难等问题的活靶子。即使躲过了这些灾难，你能躲得过腾讯吗？其实创业是好事儿，只不过各位开发者一定要先从用户的需求出发，帮用户解决一个实际的问题。这样的产品，经过几次迭代之后，还是有一线生机的。

移动互联网创业，三思而后行。

4. 程序员也可以周游世界？

一个程序员每天工作 10 小时，哪里还有时间去周游世界？但是我们程序员当中就有这么一个家伙，他辞了工作跟自己女朋友开始了自己的环球之旅。

文章作者提到了如何让技术不停歇的问题，大家可以在旅行的过程中参与开源项目，可以在旅途中带上一本技术方面的图书消磨等候的时光。当然我们 51CTO 不推荐大家都辞职去旅行，要不开发频道就无人问津了。建议大家可以好好的利用自己 2013 年的年假，与爱人、父母、子女去感受一下不同文化的氛围。

5. 谈谈 R 语言

大数据这个话题已经是所有 IT 大会必谈的科目。似乎只有 Hadoop 和 Mapreduce 才是大数据。而作为程序员的我们，是不是应该在 2013 年关注一下 R 语言？

R 是用于统计分析、绘图的语言和操作环境。R 是属于 GNU 系统的一个自由、免费、源代码开放的软件，它是一个用于统计计算和统计制图的优秀工具。相信每个程序员所在的公司都有数据量不小的数据库。我们是不是也应该学学那些所谓的数据科学家，来对自己数据库当中的用户信息进行一次挖掘？尽管它可能还不够大数据的级别。2013 年做一个数据分析方面的高手，程序员们可以试试。■

编者按

一边实习一边读书学习，花了半年考研考到帝都，在考研结束的那段时间（四个月假期）精读了数据结构，计算机组成等基础经典书籍，补习自己的基础。

从微软到谷歌-应届计算机毕业生的2012求职之路

1. 简介

毕业答辩搞定，总算可以闲一段时间，把这段求职经历写出来，也作为之前三个半月的求职的回顾。

首先说说我拿到的 offer 情况：

微软，3 面 -> 终面，搞定

百度，3 面 -> 终面，口头 offer

搜狗，2 面，悲剧

腾讯，1 面，悲剧

布丁移动，3 面，搞定

涂鸦游戏，3 面，搞定

友盟，3 面 ->CEO 面，搞定

雅虎，4 面 -> 终面，搞定

微策略，2 面，悲剧

人民搜索，3 面 -> 终面，搞定

人人，2 面 + 终面 +Special 面，搞定

Google，7 面，搞定

求职经历分为定位、准备、简历、笔试和面试这五个部分，大家挑感兴趣的看就成。

我的求职经历适用但不限于码农，不适用与企事业单位（据说是完全不同的考察标准和流程）。废话比较多，大家耐心忍受，有什么问题可以跟帖提问。

2. 定位

教育经历：本科在大连某工科院校，由于 GPA 比较惨烈 + 挂科，所以没保成研，毕业后修了一年英语双学位，然后到帝都计算机职业教育学院接受再教育。

技术能力：属于半码农半产品的类型，代码编的过去（搞过 compiler），也有一些拿的出手的产品（几十 w 的用户量），一句话描述：几十 w 代码 + 几十 w 用户的 Coder。

专业能力：非 ACM 出身，算法拙计但基础扎实。由于单身所以看了 N 多书（CS+ 心理 + 经管 + 历史），扯淡能力强大，碰到非专业的各种秒杀，碰到专业各种拙计。

实习经历：大四在一家 ds 公司实习过一年，攒了不少代码量；后来在 MS 断断续续的待了一年多，虽说是打酱油，但在众大神的光环笼罩下，水平至少提了三个档。

目标公司：由于百度给我的印象实在很差，而 MS 给我的印象又实在很好，所有就有了下面的排名：

外企（Google、MS、Yahoo 等）> 国内互联网（阿里、腾讯、百度、网易等）> 企事业单位（基本不考虑）

3. 准备

经常在论坛里看到各种求职抱怨贴，其实在抱怨前应该仔细想一想，为了求职，你付出了多少？看到人家找工作找的顺找的爽，有没有

从微软到谷歌 – 应届计算机毕业生的 2012 求职之路 II

想过人家背地里付出了多少努力和心血？别拿官二代和富二代啥的说事，真 ds 只会拿一堆自身以外的理由掩饰自己的懒惰。

不要认为求职就是发个简历等面试通知，对于大神来说不用发简历牛逼公司也会围着你转，对于 ds 来说就是预则立不预则废，中国缺什么就是不缺人，不下功夫准备很有可能连个 P 都没有。

其实很多 ds 就是怕预也废所以干脆不准备直接上，这样搞不定的话，就有借口说不是自己蠢而是自己没准备，可以捍卫自己的智商高地不被侵犯。

身边有不少这样的实例，典型的死要面子活受罪，活该你找不到工作。

我的微软 mentor 曾提到过，我的实习面试表现一般，但后来表现出的动手能力大大超出之前面试的预估，而有些面试表现很出色，问题对答如流的选手，入职之后反而不是很理想，至少没有达到面试时发挥出的水准。

这说明一个问题，就是笔试面试，准备和不准备会差异很大。如果你的简历不是那么 NB，那就只能靠笔试和面试的加分撑场面。身边经常有同学纳闷这样代码都编不利索的傻　都能进 MS 为什么我不能进，答案往往很简单 :人家比你多准备了一个月。平时电脑上写程序可能很利索，笔试面试时在纸上写写试试你就知道什么叫拙计。

IT 公司的笔试和面试的题量都不大（相对于企事业单位和银行动辄上百道选择题的题量，算是很少），一般十几道选择题，三四道大题就算题量很大。

但计算机的东西实在又是太多，程序设计、数据结构、算法设计、操作系统、体系结构、编译原理、数据库、软件工程等分支，编译的话太难（一千个码农里也没几个人能在纸上写一个最基础的递归下降 LLParser），软件工程、体系结构、数据库这些太水（不是说这些分支没用，而是它们很难考察，尤其对应届生来说这些都是些文字游戏，比如说面向对象的三要素五原则，有个鸟用），这么一排除，再把数据结构和算法设计一合并，就剩下程序设计、算法和操作系统。没错，这三项搞定，国内外 IT 公司通杀。

因此我的笔试和面试准备很简单，就是重温 + 突击程序设计、算法和操作系统。下面是我的笔试 + 面试准备内容：

程序设计：

1，把基础的数据结构的 C 语言实现在纸上写三遍以上，用我能想到的最精简最优化的方法

2，阅读 CARM 和 TCPL，确保不会遗漏 C 语言的每个细节

3，重温之前自己做过的靠谱项目，并总结里面的关键难题和解决思路

4，重　读 Writing Solid Code、Elements of Programming、Practice of programming

5，阅读 Science of Programming，做到可以证明自己的程序的正确性（前条件 + 后条件 + 不变式）

算法：

1，重读 Algorithm Design Manual，重点阅读 Dynamic Programming 和 Backtraverse

2，重读 Programming Pearls 和 More

从微软到谷歌 – 应届计算机毕业生的 2012 求职之路 III

Programming Pearls，并完成所有课后题

3，独立解决编程之美里面的题目（国内不少企业选题用的这本书）

4，完成 Careercup 里 Amazon、Google 和 Microsoft 这三个分类下面的前 20 页面试题

5，完成 TopCoder 的数十道 D1L2~D2L1 难度区间的算法题目

操作系统：

1，重读 Modern Operating System，重温 OS 的核心概念

2，重读 Computer Systems a Programmer's Perspective 的关键章节，回顾里面的关键点

从七月底开始一直到十一月，花了接近四个月，很多东西都是一边面试一边准备：面试 -> 发现盲点 -> 修复盲点。

此外列出一些面试笔试题的资源，此外感谢基友 @codewarrior 之前的推荐：

1，Crack over the code interview

很靠谱的笔试面试指导手册

2，CareerCup

集齐了大量的真实笔试面试题，去外企的一定得看

3，TopCoder

如果不是 ACM，练这个就够，其实面试也不会问太难的算法，哪怕是 google

4，编程之美

尽管题目有些过时，但依然很实用，三星题目适合一个人仔细想

此外也说下一些不靠谱的资源：

1，IT 公司面试 100 题

这个恐怕是国内传的最多的 IT 面试题

题目本身还可以，但那个出题人本身代码功底一般，给出的答案包含大量错误和缺陷，导致参考价值骤降

2，程序员面试宝典

翔一样的书，各种错误概念的堆积，如果一个错误给我一块钱，我能从这本书搞成万元户。如果去正规公司拿这本书准备，包你被黑出翔。

4. 简历

在 MS 时，老大曾让我帮忙招几个靠谱的实习生，因此我收到了几百封简历，过了一把 HR 的瘾。这里说说自己在看简历时发现的几点：

1，可读性。不要用 Word 或压缩包，用 PDF。此外在邮件里面用纯文本加上自己的简介，简化对方阅读的操作。

要记住 HR 一天看的简历海的去了，压缩包是 HR 最痛恨的格式，因为解压了就不知道扔哪去了，有时干脆就不看；Word 有版本问题，10 的 docx 到了 07 往往被黑出翔。还有就是对方有可能不在 PC 上读邮件，因此纯文本的简介非常有必要。

2，群发。不要给人群发的嫌疑，看清楚目标职位和目标公司，我发的工程院招聘贴，收到的几百封简历里面有十余封是投到微软亚洲研究院，有几个干脆写“敬爱的某领导”，尼玛这不找抽么。

3，设计。特别提一下设计，很多电工的简历就是翔，丑的一逼，对齐没有，字体拙计，要点不明。再放到几百份几千份简历里面，活该你被忽略。建议所有电工投简历前阅读■

本文未完，详细请查看：

http://developer.51cto.com/art/201301/376915.htm

面试程序员时的“唠叨”

首先，正文开始之前我先解释一下本篇文章为什么起名为面试中的“唠叨”，因为本次请到的嘉宾是就职于Amazon的程序员“四火的唠叨”（网名）。相信提起这个名字就很多人知道了，小编曾拜读过他的两篇文章，分别是《我们到底要怎样招程序员？》和《我眼中的工程师文化》，这片文章我们就说一些以上没有讲到的一些面试时的“唠叨”。

“唠叨”一下面试时经历和学历

首先，在他认为经历不能成为最为偏重的部分，原因很简单，人人都会吹牛，经历容易伪造。如此一来，面试官获得的印象，很大程度上和应聘者的口才有关系。但是，经历一定是面试中的重要组成部分。因为它可以反映出应聘者的经验和眼界，而这两者都是无法通过天赋和勤奋获得的。

如果你在一家企业文化比较类似的公司工作过，那很不错；如果你参与过一些开源项目，如果你还在社区里有知名度，那就棒极了。

经历和学历当然重要，不过我更关注应聘者“做成过”什么，而不是“参与过”什么，更不是“学习过”什么。有的人简历上经历写得满满当当，看起来很有阅历的样子，但是仔细一看发现只是隔三差五地跳槽、不断地尝试新东西而已。要做成一件可以摆上简历的事情，一定是需要潜心积累一段时间的。

到底怎样去认识对技术的“精通”

就像上述所说的，简历可以造假，而且在简历中经常见到的一个词就是“精通”，尤其是在刚毕业的学生；刚从培训基地出来的学生；没有多久工作经验的程序员简历上屡见不鲜，“精通”这个词并不会给你赢来多少好感。

要对某种语言“精通”其实并不容易，许多人觉得语法掌握、运用自如就可以称为精通了。其实语法语义只是最粗浅的层面而已，精通一门语言，还需要使用它的类库，理解它的优势和缺陷，了解它的平台和实现，甚至包括它的发展演进过程等等。在简历上，说出你使用某项技术，做出了什么东西，要比填写“精通XXXX”有意义得多。当然，简历只能作为参考，只有面试中的表现才能令人信服，即便面试总会有片面和运气的成分。

在这一点上，跟大家“唠叨”唠叨，很多人都去过培训机构这种地方，小编也曾经去过，无论哪老师总会爱重复一句没用的话就是：‘你们以后月薪最低3K5’，还在最后在简历中叫你如何熟练的去应用“精通”这一词……

不巧，“四火”也曾经参加过类似的培训班，老师也很能说，可是他并没有学到什么扎实的东西。中国培训机构的口碑众所周知，以至于在简历上写“XX培训机构毕业”兴许会掉价的，呵呵。

我让你进入团队的原因并不单单是技术

这段开始之前，小编先引入《我们到底要怎样招程序员？》中的一个段落。看过的朋友可以跳过继续往下看。

‘招合适的人，码农也好、程序员也好，刺儿头也好、老好人也好

这又是理想和现实之间的平衡，我以前的

面试程序员时的“唠叨”II

一位朋友招人的时候，就明确了希望寻找听话的人、寻找毕业不久的孩子，“可以不很优秀，但是可以让我不花太多精力在监督他们的工作上”。这就像一些朋友给我的留言所说的一样，中国太需要程序员了，但是中国的很多企业目前也非常需要码农，很容易洗脑、很容易管理，努力干活，又不闹事儿，你不能指望他们能干出什么伟大的事儿来，但是他们不给你添乱，老老实实的干活……这是多么残酷、多么讽刺的话，但是无比现实，尽管我希望这样的场景越少越好。

对于刺儿头，我还真遇到过一个相当典型的，那是在我刚参加工作的时候，只要有争论，就见其找茬，到后来我们开会都不敢带着她，一个刺儿头的破坏力，远远大过几名优秀程序员给团队带来的正面影响。但是团队中，又不能都是老好人，这样的团队会缺少活跃的气氛、缺少想法，甚至缺少做决策的人。’

小编和唠叨的对话中问道若是他的团队需要融入一个新人时，更希望引入一个上面样的人，是老实人、刺头、或是可以有一种新的形式 :相对老实的刺头。

他的回答是招人首要的因素还不是技术和经验，而是他是否认可你的团队文化，他是否可以融入这种文化。

在很多公司，面试录取之后，是由特定的人将录用者安插到不同团队中的，程序员并没有选择权。

其实他特别反对这样的做法。尤其在小团队中，面试过程一定要有自己团队的人参与，而且这个比重不能太小。原因是，应聘者有可能是和你从早到晚一起工作、生活，是你的好伙伴、好基友，一定要让足够的选择权掌握在自己团队手里，以让大家在未来合作得愉快。

这也是我坚持认为要让程序员去面试程序员的一个原因，老实人也好、刺儿头也好，团队总是具有包容性的，只要彼此认可，没有什么不可以。

记忆犹新的中国式程序员

两年以前，曾经遇到过这样一位应聘者，他的能力和技术背景也符合我们的要求，但是当时我们的项目比较辛苦，就和他说，可能会经常加班，结果他表现出了一副“工作狂”的态度，表示愿意经常吃住在公司 ;问他喜欢做什么方面的项目，他的回答是“愿意无条件服从分配”……这件事情对我触动很大，后来发现类似的情形居然不在少数。从那以后我常常思考，到底是什么让中国的程序员变得如此饥不择食?

希望有一天，中国的程序员们都可以拍着胸脯介绍自己，自豪而且坚决。而我，很想多做一些事，去帮助实现那个伟大愿望。这也是我写博客的一个目的。

和你唠叨一下如何找下家

面试不只是面试官考察应聘者，同时也是应聘者在考察团队和工作，这是一个双向的过程。但是找下家你要认清自己，认清公司。

★ 以后要做的事情，是不是够有意思?

★ 未来的职业发展，是不是能满足自己的野心?

★ 和自己打交道的人，是不是可以很好相处?

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201301/376714.htm

我们是否正确看待了跳槽这件事

跳槽在任何行业都是屡见不鲜的一种现象，尤其是在年关将近，年终奖即将发放的此时，很多人都开始去寻找新的下家，为年后跳槽做准备。跳槽后出现的情况无非就是‘我现在过得比以前好’和‘还是以前的工作好’。当你出现第二种情况的时候，说明你并没有理智的去看清你所在的行业，只是盲目的看到了比你过得好的那些人，盲目的选择了跳槽。

这次很荣幸邀请到了《谈谈离职和跳槽》这篇文章的作者张子阳张老师，身为技术部部门经理的他对这个问题也有着较为深入的思考，接下来我们就去看看跳槽这件事。

追求和目标

有些人选择了留在发展较好的小公司；有些人留在了看似不错的大公司平淡的混了很久；还有人离开了发展较好的小公司去选择一个发展较慢但相对上家公司较大的平台。那么我们应该通过什么来认识到自身的价值来规划自己的发展？有些跳槽是否又是盲目的呢？

张老师给出的回答是这样的：“我觉得这个和一个人的追求和目标有关的。”

有些人就喜欢平平淡淡、安稳的日子，那么待在一个看似不错的大公司，上班时把手头的工作忙完，下班了就休息娱乐，也没什么不好；有些人就一定要自己做老板，宁可摆个地摊也不去打工，宁为鸡头不为牛后。根据自己的能力和背景，有可能成功有可能失败，但是也没什么问题；有些人就喜欢不断努力，朝着社会精英阶层奋斗，那么他就不断地磨练自己，当发现一个地方对自己的成长起不到太大帮助的时候，就会选择离开，这样也是值得鼓励的。

至于说的很多人盲目地选择了离开公司，我觉得不能说盲目的，前途好不好不是老板说了算的，这个主要是来源于要离职的当事人自己的判断。但是不管在任何情况下，离职都要进行一个充分的思考。在想要离职时，最好写一篇总结，标题就叫《我为什么离开 XXX》。当完成了这篇总结后，如果还犹豫不决，就请请你的长辈，请你的同学、朋友、爱人看看，看看有没有道理。当然，他们的意见只能作为参考，只是让自己能够听到不同的声音，决定权还在自己。

中国人有个传统情结，叫做“劝和不劝离”，这句话虽然是在说夫妻感情，但放在公司上也是一样的。而且大多数人都是“风险厌恶型”，因此大多数人都会劝你留下，你要有自己的判断。

所在的平台

在《谈谈离职和跳槽》中提到张老师是在一个不大的平台，通过 51CTO 记者的了解现在他也还在文章中提到的这个平台，在访谈过程中他叙述了他考虑到的几个方面：‘对我来说，我觉得在一家公司待下去，基本上有三个出路：技术专家、管理专家、行业专家。

首先，相对于其管理专家和行业专家，我对于技术专家的兴趣相对低一些。因为，如果仅仅是你一个人的技术很出众，那么你能成就的事情是很少的。

我们是否正确看待了跳槽这件事 II

举个例子，你是一个能工巧匠，可以雕刻出造型最精美的石像，可问题是产量有限，而产量有限的话你的价值也就有限。当然，你可以通过你的技术能力去带领其他人，创造更大的价值。可是，这时候你就不再是一个人了，不再是纯粹的一个技术牛人，你要带领一个团队，而带领一个团队时，你就可能需要成为管理专家。我相信没有哪家公司的 CTO 是纯粹做技术搞研发的，他应该是具有相当深厚的技术功底，同时又具备一定的管理能力。

从技术专家的角度来看，这家公司并非技术型公司，如我之前文章中所说，5 年工作经验就够了，那么从技术上来说，我的成长是有限的。我还是比较在乎自己的技术水平的，而且也在不断充电，我不一定要花太多时间钻研一些奇技淫巧，但某个技术可以用来做什么有什么特点我还是要去了解的，毕竟我也还是一个技术人员，而且和我打交道的最多就是工程师。对技术掌握得越多，沟通起来障碍就越少。

再说一下管理专家，因为公司的规模不大，所以尽管我所在的技术部门只有五个人，但我的顶头上司就只有老板一个人了。平心而论，部门内的人基本上都能够做到尽心尽力、尽职尽责，即使是即将要离开的同事，也依然积极地在为公司做事，没有流露出什么负面情绪，大家相处的也都很好。而公司里跟技术相关的重要系统、产品，不是已经翻新重做了，就是重大升级了。

对于管理专家来说 :最重要的就是从纷繁复杂的事务中整理出头绪，分清楚哪些是重要的事情、哪些是紧急的事情、哪些是次要的事情，然后依照紧急重要的次序，有条不紊地完成。同时，要能发挥下属的潜能，尽量做到让大家齐心协力朝一个方向前进。

最后说一下行业专家，很多时候行业专家都是公司的老总，正因为他对这个行业的了解，最终才决定成立公司，在这个行业开疆拓土。如果只是想当一个程序员，很多时候在哪个行业是没有多大关系的。反正在这家公司也是写程序、做系统，去那家公司也是写程序、做系统，有什么关系呢？但是如果往更高层去的话就不一样了。其实就算是程序员，在做一个行业的系统时，也会成为这个行业的半个专家，因为你要去了解流程，你的系统究竟是应用于什么场景，解决哪些问题？当你在一家公司做久了以后，就会倾向于成长为这个行业的行业专家。但是各个行业的规模是不同的，各个行业所能创造的价值也是不同的。我现在所处这家公司所在的行业，防伪防窜货，整个行业中一家上市公司都没有。可见，这个行业是很小的，能创造的价值也是有限的。所以，就需要考虑一下，十年以后，究竟是成为一个防伪防窜货专家、一个电子商务专家、一个移动互联网专家、一个物联网专家还是一个 ERP 专家？举一个极端点的例子看得更清楚一些，一个通下水管道的专家和一个智能手机行业的专家，所能创造的价值大小是不同的，而一个人的价值和成就是以他为社会所做出的贡献来衡量的。那么如果想有更大的成就，那么就需要选择一个能创造更大价值的行业。当然，这样的行业能力强的人也多，竞争也激烈，我想除了谦虚、勤勉以外没有更好的办法了。■

本文未完，完整部分请浏览
http://developer.51cto.com/art/201301/377155.htm

我优化YouTube视频播放页的故事

三年前，当我还在 YouTube 做一名 web 程序员时，有一位资深的工程师发了一通牢骚，说播放视频的页面体积太大。

三年前，当我还在 YouTube 做一名 web 程序员时，有一位资深的工程师发了一通牢骚，说播放视频的页面体积太大。这个页面体积已经膨胀到了高达 1.2MB，包含有数十次的请求。这个工程师公开的宣称说，“如果他们 Quake 能在 100KB 的体积下克隆出我们的页面，我们没有理由达不到这个体 积！”因为我同意他的观点，并且我正在找新的任务，于是就决定接受这个挑战，让 YouTube 的视频播放页面的体积减到 100KB 以下。

那天晚上从旧金山回家的火车上，我编写了一个基本的原型。我决定限制页面上的功能数，只保留一个标题，一个视频播放器，五个相关视频，一个分享按钮，一个插旗工具和十条评论——是通过 AJAX 加载的。我把这个任务命名为“羽毛”。

即使这样一个有限的功能，页面的体积仍然达到 250KB。我深入代码查看，发现我们的优化工具（比如闭包编译工具）无法清理这个页面上实际没有使用 的代码（也许不该责备这些工具，这种情况下任何工具都做不到）。

想进一步减少代码，唯一的方法就是手工优化 CSS，JavaScript 和图片。经过辛苦 的三天努力，我已经把页面做到了相当的精瘦，但仍然没有低于 100KB。因为我刚刚写完了一个 HTML5 视频播放器，我决定用它来替换体积笨重的 Flash 播放器。砰！ 98KB，只有 14 个请求。对这个页面设置了一些基本监视后，我们对一小部分人开放了这个页面。

经过了一周数据的收集，数据有了，但它们却让我困惑。羽毛版下的页面的总体平均延迟时间实际上是增加的。我减少了总的页面体积，减少了页面请求的次数，但数据显示在加载羽毛视频播放页却花了更长的时间。这是不可能的事情。

深入挖掘数据，经过在浏览器上的反复试验，没有任何结果。我基本上要放弃这个版本了，我的信仰几乎被完全击溃，正在此时，一个同事发现了其中的奥秘 :地理因素。

当我们标注了数据的地理信息，把所有信息按区域划分进行对比，我们看到了地区，比如东南亚，南美，非洲，甚至西伯利亚等地在流量上呈现的不对称增加。进一步调查揭示，在这些地区，羽毛版的页面的平均加载时间超过 2 分钟！这意味着，一个普通的视频，大概 1 兆左右，会需要 20 分钟来加载！人们为了等待这个页面就如此痛苦，更别提视频了。可纵观这些地区，他们之前根本无法观看 YouTube，因为等了很久也看不到什么。而在羽毛版下，尽管要等 2 分钟才能看到视频的第一帧，但不管怎样，事实上是可以看到了。在过去的一周里，羽毛版在这个地区很受欢迎，所以我们的数据被他们弄的完全不平均了。大量以前不能观看 YouTube 的人现在突然可以了。■

编者按

我翻译此文并非推崇 C 而贬低其他语言。我翻译此文，只是因为作者的多处精到的见解让人深思。作者的出发点，很明显，是纯技术的；各位读者且谨记这一点。

C语言高效得简直不合理

多年来，我一直试图摆脱 C 语言。太简单，太多细节需要处理，太古老，太低级。我一直钟爱 Java，C++，Erlang。我用它们创建了很多项目，并且自己为这些项目感到骄傲；然而，这些语言，最终，都伤了我的心。

他们做出承诺，却无法兑现；他们专注于错误的东西，并且所做的“折衷”最终让你倍感煎熬。于是，我不得不求助于 C。

C 就是一个万能背包。它高效且高产，有强大的工具和广泛的社区支持，并且它对它所做的“折衷”非常诚实。

对于其他语言，他们能让你更快的工作，但从长远来看，当性能和可靠性变得重要时，C 将会为你省去不少麻烦事儿。我个人再次非常痛苦的学到了这一刻。

简单直观

C 语言是非常棒的高级语言。我重复一遍，C 语言是非常棒的高级语言。当然，它没有 Java、C# 等高级，自然也没有 Erlang、Python 或者 Javascript 高级。

但是，他和 C++ 在语言的高级程度上，是一样的；而然它比 C++ 更加简单。当然 C++ 提供了更多的抽象，然而它并没有给出比 C 更高级的抽象。在使用 C++ 时，你考虑的细节并不比你使用 C 时的少，除此之外，你还要考虑一堆可笑的无意义东西。

> "When someone says: 'I want a programming language in which I need only say what I wish done', give him a lollipop." – Alan J. Perlis
>
> 当有人说：“我想要一种编程语言，我仅需要对它说我想干啥就行了。”那么给那个小屁孩儿一个棒棒糖吧。 Alan J. Perlis

我们想要找一种低级语言来代替 C，然而找不到；这并非是因为 C 语言是低级语言，相反，恰恰是因为 C 语言作为底层机器上的高层抽象太成功了。它如此成功，以至于让大多数的低级语言显得毫无意义。C 就是这么擅长它所做的。

C 语言的语法和语义强大而直观。它可以用以编写高级算法，同时也可以用以处理底层硬件逻辑。正因为其强大、简单和直观的语法和语义，C 语言并不会给我们一些额外的认知上的负担，从而让编程者专注于真正重要的事情。

C 颠覆了我们对低级语言的认识。这真了不起。

简单的代码，精致的类型

c 语言是一种弱类型语言，其类型系统非常简单。和 C++ 还有 java 明显的一个区别是，c 里面你不能定义“类”（class），你不可以把所有的运行时需要的东西都放到“类”里面。

C 语言高效得简直不合理 II

你的所有工作都严格基于结构(struct)和联合(union)。所有的函数调用者必须明确被调用函数的参数类型和返回值类型。所以调用者的自由相对有限。

你只是想要个香蕉,结果来了只自称森林之王的大猩猩——Joe Armstrong

你刚刚听起来像是 c 语言缺点的东西某种程度上确实一种优点 :c 语言的 API 面对用户都力图精简。这避免了庞杂的框架,而力图在简单的类型基础上创造一个小巧的函数库。

而面向对象的语言往往在复杂的类型基础上又构造了庞杂的基础类库,这些库提供了大量的相互依赖的接口,他们的参数和返回值的“类”型也因此更加复杂。每一种“类”又定义了大量的复杂的方法和属性……好吧,更加复杂了。

这并不是说吐面向对象就希望变复杂,但是他们貌似鼓励你把事情变复杂。他们的复杂性使你很容易犯错误。相对来说, c 就很少导致错误。c 语言尽力构建一个简洁、通俗的类型系统,使用它你会发现你不需要顾及那么多的依赖关系。这使你的开发变得更加简单。

速度之王

c 语言不论在处理器中还是在内存堆栈里,都是速度最快的。而且其高效不仅仅体现在速度上,即使是内存的管理以及启动时间上,也无人望其项背。当你需要平衡空间和时间的消费时, c 语言从来不会对你隐藏任何细节,原因如下 :

– 强大的编译器

–k&p 风格

每次那些更高层次的编程语言(比如 java 或者 haskell),声称自己能产生接近 c 语言的表现从程序的时候,这在我听来简直就是笑话。通常,他们为了实现这一点,不得不在语法上做出一些稀奇古怪的事情,比如专门搞一些“聪明的”编译器或者虚拟机……这种古怪的优化行为使语言失去了原本简单的性质,更何况这种优化往往只是针对处理器

当你想要用 c 语言写一些对运行速度要求严格的东西时,你可以很清楚的知道为什么他很快,这一点不因为你使用的编译器或者虚拟机不同而改变。应用程序中, GC (垃圾回收)的设置将会影响运行。而人机交互将会影响垃圾回收对于数据的处理。

c 语言的代码优化直接而有效。即使你不这样认为,在实际工作中也有大量的工具帮助你了解其中的缘故。相对来说,你根本没有必要为此壮起胆子去尝试学习什么虚拟机,什么“智能优化编译器”。当你在使用 cpu,内存和 IO 分析器的时候, c 语言绝对不会让你对底层到底发生了什么感到困惑。以上所言,不论是从处理器的角度还是从内存堆栈角度,都证明了 c 语言是速度之王。

更快的“编写 – 运行 – 调试”周期

“编写 – 运行 – 调试”这个开发周期对于程序员是十分重要的。如果这个周期足够快,开发中的人机互动足够多,那么你的任务就进行的足够迅速。■本文未完,完整部分请浏览

http://developer.51cto.com/art/201301/376894.htm

我是怎样教媳妇面向对象编程的

■ 简介

由于我在软件开发的经验，我总是发现无论一个技术问题看上去多么难搞，只要从现实生活的角度去解释或用对话的方式去讨论总能让它变得更简单。

我老婆 Farhana 想要继续软件开发生涯（之前因为我们的第一个孩子出生，她不得不放弃）。我已经有了一些软件设计和开发的经验，所以这几天我就在试着帮助她学习 OOD。

由于我早年在软件开发的经验，我总是发现无论一个技术问题看上去多么难搞，只要从现实生活的角度去解释或用对话的方式去讨论总能让它变得更简单。关于 OOD，我们已经有了许多成果丰硕的讨论，我觉得有人可能发现这是一个学习 OOD 有趣的方式，所以我想我应该分享出来。

下面是我们的谈话步骤：

话题：介绍面向对象设计

丈夫：亲爱的，让我们开始学习面向对象设计。你了解面向对象规范，对吗？

妻子：你是指封装，继承和多态吗？是的，我了解这些规范。

丈夫：行，我想你已经知道怎么用类和对象了。今天我们来学习面向对象设计。

妻子：等等。了解面向对象规范对面向对象编程来说难道不够吗？我的意思是，我能够定义类，封装属性和方法。我能够根据它们的关系定义类的继承。那还有什么呢？

丈夫：很好的问题。面向对象规范和面向对象编程完全是两码事。让我展示一个现实生活中的例子来帮助你理解它们。

我们从牙牙学语起，都是先从字母表学起的，对吧？

妻子：嗯。

丈夫：好，然后你就能认单词了，还能通过不同的字母拼写出不同的单词来。慢慢的，你能通过一些基本的语法把这些单词串成一句话。为了使句子时态正确且没有语病，你需要用一些介词，连词，等等。。看下面这句话

"I"（代词）"want"（动词）"to"（介词）"learn"（动词）"OOD"（名词）

通过把几个单词摆放妥当一句话就好了，然后用个关键词来说明一下这句话的重点。

妻子：亲爱的，你闲扯这些到底要说明什么呢

丈夫：我说的这个例子跟面向对象规范很类似，面向对象规范为面向对象编程定义了基本的规范，它是面向对象编程的主要思想。面向对象规范好比基本的英语语法，这些语法教会了你怎么用一个个单词拼凑出一句句话来，而面向对象规范教你怎么用类，怎么把一些属性和方法封装在一个类里，怎么串出类之间的继承关系。

妻子：啊哈，我知道了，那么，面向对象适用于哪里呢。

丈夫：听我慢慢道来。现在，假设你想写点有内容有题材的文章。你当然还希望写点你比较擅长的题材的书，就会简单造几个句子是远远不够的，对吧。

我是怎样教媳妇面向对象编程的 II

你需要笔耕不辍写出一些长篇大论，你还需要学习怎么可以让读者很容易就看懂你写的这些长篇大论。。。

妻子：嗯，有那么点意思。。。继续吧

丈夫：现在，假如你想写本关于面向对象设计的书，你需要把这个大的课题拆分成一些小题目。把这些小题目分几个章节写，还得写前言，简介，说明，举例，一篇里还有很多段落。你需要设计一整本书，还得练习一些写作技巧，让文章读起来浅显易懂。这就是综观全局。

在软件开发中，OOD 就是用来解决从全局出发考虑问题，在设计软件的时候，类和代码可以模块化，可重复使用，可灵活应用，现在已经有很多前人总结出的类和对象的设计原理了，我们直接拿来用就行了。

总之，历史的车轮已经碾压出一条清晰的车轮印，我们只要照着走就可以了。

妻子：哎，懂了点皮毛，还有很多要学呢。

丈夫：不用担心，你很快就会上手的，让我们接着来吧。

话题：为什么要进行面向对象设计？

作者：有个很重要的问题，既然我们能够很快的创建几个类，编写程序并提交，为什么我们还要关注面向对象设计？这样不够么？

妻子：恩，以前我不知道面向对象设计，我也能开发提交项目。有什么关系？

丈夫：好吧，先让我给你看一个经典的引述：

"需求不变的程序开发会同行走在冰上一样简单。" – Edward V. Berard

妻子：你指软件开发说明书会被不断修改？

丈夫：非常正确！软件开发唯一的真理是“软件必然修改”。为什么？

要知道，你的软件解决的是现实世界中的问题，而现实生活不是一成不变的。

可能你的软件现在运行良好。但它能灵活的支持“变化”吗？如果不能，那它就不是一个敏捷设计的软件。

妻子：好，那你就解释一下什么叫做“敏捷设计的软件”！

丈夫：“一个敏捷设计的软件能轻松应对变化，能被扩展和复用。”

而应用“面向对象设计”是做到敏捷设计的关键。那么，什么时候你可以说你的程序应用了面向对象设计？

妻子：我也正想问呢。

丈夫：如果代码符合以下几点，那么你就在“面向对象设计”：

面向对象；复用；变化的代价极小；无需改代码即可扩展

妻子：然后呢？

丈夫：不只我们。很多人也花了很多时间和精力思考这个问题上，他们尝试更好的进行“面向对象设计”，并为“面向对象设计”指出几条基本的原则（你可以用在你的“面向对象设计”中）。他们也确实总结出了一些通用的设计模式（基于基本的原则）。

妻子：你能说出一些吗？

丈夫：没问题。现在有许多设计原则，但是最基本的，就是 SOLID（缩写），这五项原则。（感谢鲍勃叔叔，伟大 OOD 导师）。

我是怎样教媳妇面向对象编程的 III

S = 单一责任原则

O = 开闭原则

L = Liscov 替换原则

I = 接口隔离原则

D = 依赖倒置原则

在下面的讨论中,我们将详细了解这些。

话题 :单一功能原则

作者 :让我们先来看图,我们应该感谢制作这张图的人,因为它们真的太有趣了。

单一功能原则图

它的意思是 :"如果你可以在一个设备中实现所有的功能,你却不能这样做"。

为什么呢? 因为从长远来看它增加了很多的可管理性问题。

从面向对象角度解释是 :

" 导致类变化的因素永远不要多于一个。"

或者换行个说法 :" 一个类有且只有一个职责 "。

妻子 :可以解释一下么?

丈夫 :当然,这个原则是说,如果有多于一个原因会导致你的类改变 (或者它的职责多余一个),你就需要根据其职责把这个类拆分多个类。

妻子 :嗯 ... 这是不是意味着在一个类里不能有多个方法?

丈夫 :当然不是。你当然可以在一个类中包含多个方法。问题是,他们都是为了一个目的。那么,为什么拆分很重要的?

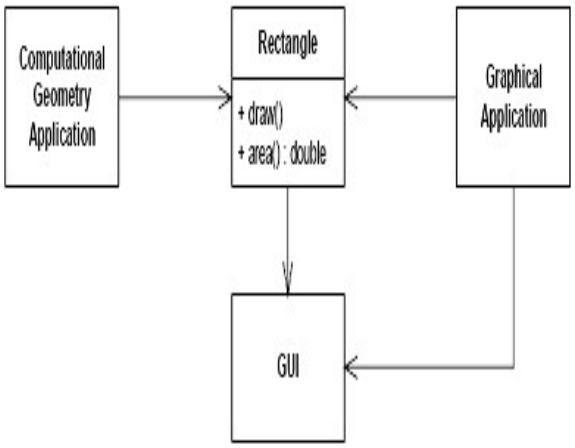
那是因为 :

每个职责都是轴向变化 ;

如果类包含多个职责,代码会变得耦合 ;

妻子 :给个例子呗?

丈夫 :木有问题啊,瞅瞅下面类的结构。其实,这个例子是 Bob 叔叔那儿来的,得谢谢他。



违反 SRP 原则的类层次结构

这里,Rectangle 类干了下面两件事 :

计算矩形面积 ; 在界面上绘制矩形 ; 而且,有两个程序使用了 Rectangle 类 :

计算几何应用程序用这个类计算面积 ; 图形程序用这个类在界面上绘制矩形 ;

■本文未完,完整部分请浏览
http://developer.51cto.com/art/201301/376735.htm

如何向妻子解释设计模式

自上篇翻译 < 如何向妻子解释 OOD> 后收到了很好的反应。故特继续翻译作者的 <How I explained Design Patterns to my wife: Part 1> 一文，以飨读者。在此文中，作者依旧通过与妻子浅显易懂的对话，向读者解释了什么是设计模式。

设计模式是什么？

Shubho: 通过我们关于面向对象设计原则（OODP，即 SOLID 原则）的对话，我想你已经对面向对象设计原则（OODP）有了基本的认识。希望你不要介意我把对话分享到博客上。

设计模式是这些原则在某些特定公共场景下标准化的应用，接下来让我们通过一些例子学习什么是设计模式。

Farhana: 当然，我喜欢例子。

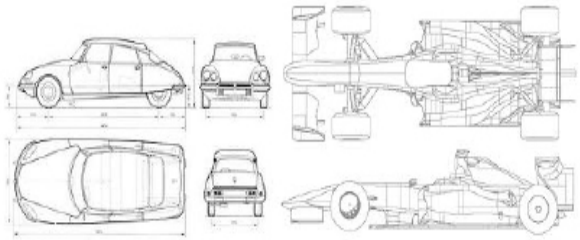
Shubho: 让我们以汽车为例讨论一下。汽车是一个很复杂的对象，由成千上万的其它对象组成，如发动机，车轮，方向盘，车座，车体等等其他不同的部分或部件。

当装配汽车时，制造商需要集中并装配这些更小的自成汽车子系统的不同部件。而这些不同的小部件同样也是复杂的对象，其它制造商同样要生产并组装它们。在生产汽车时，汽车公司并不会为怎么生产组装这些部件操心（前提是他们要确保这些对象 / 设备的质量）。当然，汽车制造商更加关心怎么装配这些不同部件以便能生产不同型号的汽车。

Farhana: 汽车制造公司必须有如何生产不同型号汽车的设计图或蓝图，对吗？

Shubho: 当然，并且这些设计都是良好的，他们花费大量的时间和精力来做这些设计。一旦设计完成，生产汽车就仅仅是照葫芦画瓢了。

Farhana: 嗯。如果事先有一些好的设计，就能在短时间内遵照这些设计生产不同产品，并且制造商在每次生产某一个型号产品时就不需要重新设计或重新发明车轮，他们只需要按照已有的设计办事就行了。



Shubho: 你抓到重点了。现在假设我们是软件生产商，我们使用基于需求而来的不同组件或功能构建各种不同的软件程序。当生产这些不同软件系统时，我们常常需要为一些不同软件系统中存在的相同情况开发代码，对吗？

Farhana: 是的，在开发不同软件程序时经常遇到相同的设计问题。

Shubho: 我们尝试使用面向对象的方式开发软件，并尝试应用 OOPD 来让代码能易于维护，可复用，可扩展。

无论什么时候，当我们遇到这些设计问题时，如果我们有一组经过谨慎开发，良好测试的对象以供使用会不会更好呢？

Farhana: 是的，这样能够节省时间，生产出更好的软件，且利于以后维护。

如何向妻子解释设计模式 II

Shubho: 很好！从设计上来说，它的好处是你不需要开发那些对象。经过多年发展，人们已经遇到过一些类似的设计问题，并已经形成有一些公认的，良好的已标准化的设计方案。我们称之为设计模式。

我们一定好感谢四人组，他们在《设计模式：可复用面向对象软件设计》中总结出了 23 种基本的设计模式。四人组由 Erich Gamma, Richard Helm, Ralph Johnson, 和 John Vlissides 组成。实际中有很多面向对象设计模式，但这 23 种模式被公认为是所有其他设计模式的基础。

Farhana: 我能发明一个新的模式吗？这可能吗？

Shubho: 当然，亲爱的，为什么不能呢?！设计模式不是由科学家发明创造的。它们是被发现找到的。这意味着任何通用问题场景中都有一些好的设计方案在那。如果我们能够指出一个能够解决一个新的设计相关问题的面向对象设计，那么这将会是一个由我们定义的新的设计模式。谁知道呢?！如果我们发现找到一些设计模式，或许将来有一天人们会称我们为二人组，哈哈。

Fahana: :)

我们将如何学习设计模式?

Shubho: 我一直认为例子是学习的最好途径。在我们的学习方法中，我们不会先讨论理论后讨论实现。我认为这是很糟糕的方式。设计模式不是基于理论的发明。事实上，问题场景首先出现，其次是基于这些问题的来龙去脉和需求，然后是一些设计方案的演化，最后其中的一些被标准化为模式。所以对每一个我们讨论的设计模式，我们将尝试理解并分析一些现实生活中的例子，然后一步步尝试归纳一个设计，并最后总结一些与某些模式匹配设计。设计模式就是在这些相似过程中发现的。你认为呢?

Farhana: 我想这种方式对我更有用。如果我能通过分析问题和归纳方案得出设计模式，我就不用死记那些设计模式和定义了。请按照你的方式继续。

一个常见的设计问题和它的解决方案

Shubho: 让我们考虑下面的场景：

我们房间里有些电器(电灯，风扇等)。这些设备按照某些方式布局，并由开关控制。任何时候你都能替换或排查一个电器而不用碰到其他东西。

例如，你可以换一个电灯而不需要换开关。同样，你可以换一个开关或排查它而不需要碰到或替换相应的电灯或风扇；甚至你可以用把电灯连接到风扇的开关上，把风扇连到电灯的开关上，而不需要碰到开关。

Farhana: 是的，但就是这样子，对吗?

Shubho: 是的，确实如此，就该如此布局。当不同东西联系在一起时，它们应该按照一定方式联系：

修改或替换一个系统时不会影响到另一个，或者说即便有，也应该最小化。这能够让你的系统易于管理，且成本低。想想一下，如果改一下房间里的灯同时需要改开关，你会乐意在你房子上花钱并安装这个系统吗?

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201301/376831.htm

看看JDK 8 给我们带来什么

世界的变化虽然缓慢但一直在变。继 JDK 7 给 java 一个全新的面貌之后，java 社区就一直期盼着 java 剩余的全部改进空间可以伴随着 JDK 8 甚至很可能是 JDK 9 的诞生而消失。JDK 8 的目标是填补 JDK 7 存在的实现空白 – 部分遗留的难以实现的问题，在 2013 年底广大的开发者将可以从三个具体的方向改善和提高这门语言：

开发效率 ;性能 ;模块化

因此，从明年开始，java 会通过在各个平台运行的方式（手机，云端，桌面，服务器等）来作为优化改进的一种方式。接下来，我将把 2013 年我们所期待的做一个概述——恰好该做新年年度计划——之后，我们将把重点放在提高开发效率的 lambda 项目上，以及在编写代码中如何将 lambda 表达式引进。

开发效率

生产效率方面 JDK8 主要从以下 2 个目标提升：

– 集合（ collections ）– 通过对集合扩展，让使用时更加简洁

– 注解（ annotations ）– 加强注解支持，允许在上下文中写注解，现在是不能这样用的（如：primitives ）

把 Fork/Join 框架加到 JDK7 中，是我们转向多核编程的第一步。JDK8 通过提供闭包（ lambda 表达式）支持的方式将这条路线走的更远了。可能影响较大的就是集合部分吧，闭包再加上新的接口和功能将推使 java 容器到一个新的层次。除了更加增加可读性和代码的简洁性，lambda 表达式还使集合操作能充分利用多核处理器特性。

模块化

社区中最让人感兴趣的一块是 jigsaw 项目：这个项目的目的是为 JAVA SE 平台设计和实现一个标准模块化的系统，然后把这个系统应用到平台本身和 JDK。这里我用了过去式的说法是为了那些我们希望摆脱类路径（环境变量）和类载入器，我们不得不把期待留到 JAVA9，至于那个时间点，也会因为 jigsaw 项目而被推迟。

我们来看一下 2013 年 java 的里程碑：

2013/01/31 M6 功能完成

2013/02/21 M7 开发者预览版本

2013/07/05 M8 最终候选版本

2013/09/09 GA 通用版

除了 jigsaw 项目，另外一个让我们兴奋的大变动（在这个版本）将要到来，那就是闭包的支持！在 lambda 表达式的帮助下，jdk 将有了关键性的提升。

Lambdas

首先，我们需要下载个支持 lambda 的 jdk，有两种方式可以获取到：

⋆ 一个用于敢于尝试的人 :从 sources 源码自己构建

⋆ 快捷版 :直接下载编译好的 sdk

最初，我使用源码构建，但由于时间原因，再加上和环境变量有关的一些警告，我选择了偷懒的方法 :使用已经构建好的 jdk。

看看 JDK 8 给我们带来什么 II

另外一个重要的工具，是文本编辑器用它来写代码。在以前，jdk 刚发布后一段时间，一个支持的 IDE 才产生。但这次不同了，也可能由于 openjdk 提供的透明和应用广泛的 jdk 有关。几天前，JetBrain 第一个支持 java8 的 IDE 发布了。因此，IntelliJ IDEA 12 成了第一个支持 JDK8 的 IDE，除此之外的改进呢？处于测试目的，我在 win7 x64 机器上安装了支持 jdk8 b68 的 IntelliJ 12 社区版本。那些喜欢 Netbeans 的开发者，可以猛戳此处 download 下载对 lambda 支持的包。

调整到合适的心态

想要尝试运用最新的特性编写出更加高效和整洁的代码，你必须了解一下几个新的概念 -- 好吧，至少鄙人需要。什么是 lambda 表达式？

最简单的看待 lambda 表达式的方式就是，你可以把它看做一个方法："它提供一系列的正式的参数和一个通过这些参数来表述逻辑的方法体 --- 它可以是一个表达式或者一个代码段。lambda 表达式的参数可以是声名的或者引用的，当这些参数是引用类型的时候，那么这些类型就是源于针对 lambda 表达式的功能性接口。从返回值来看，一个 lambda 表达式可以是无返回值的 -- 它们不返回任何结果，或者是有返回值的 -- 在表达式里面的某个执行语句返回一个值。

什么是功能性接口呢？一个功能性接口就是一个只含有抽象方法的接口，只是声名了一个函数。在某些场合下，这个唯一的函数可能是一个带有重载因子的的多态函数，这种情况下，所有的函数对外都是一个函数。除了典型的通过新建和初始化一个类来新建一个接口实例，功能性接口实例还可以通过使用一个 lambda 表达式、方法、或者构造引用来达到新建实例的效果。下面是一个功能性接口的例子：

```
// custom built functional interface
public interface FuncInterface {
  public void invoke(String s1, String s2);
}
```

下面是来自 java api 的功能性接口：

```
java.lang.Comparable
java.lang.Runnable
java.util.concurrent.Callable
java.awt.event.ActionListener
```

接下来让我们来看看一个线程的启动在 future 中可能会发生怎么的变化。旧方式：

```
new Thread(new Runnable() {
        @Override
        public void run() {
          for (int i=0; i< 9; i++) {
                System.out.println(String.
format("Message #%d from inside the thread!", i));
          }
        }
      }).start();
```

■本文未完，完整部分请浏览
http://developer.51cto.com/art/201301/376556.htm

回首与期待，JavaScript这一年

2012 年，JavaScript 总体态势很好，不像 HTML5 一直处在风口浪尖。也未受到其它语言（Dart，CoffeeScript）的影响，仍然是开发者们最喜爱的前端编程语言。

Web 技术每年都在日新月异的变化着，虽然这样，但仍然有一些语言处于屹立不倒的位置，比如本文要讨论的 JavaScript。

JavaScript 自 1995 年诞生以来已过去 17 个年头，它被广泛地应用在 Web 开发中，随着 HTML5 技术的发展，JavaScript 在未来还将有更大的发展和应用空间。JavaScript 从过去“装饰性”的一种脚本语言转变为主流的编程语言，例如在 2012 年 3 月份发布的编程语言排行榜中，JavaScript 占据第 8 名，超越了 Perl 和 Python。

依旧春光明媚

2012 年，JavaScript 总体态势很好，不像 HTML5 一直处在风口浪尖。也未受到其它语言（Dart，CoffeeScript）的影响，仍然是开发者们最喜爱的前端编程语言。

那么下面让我们来简单的分析一下 JavaScript 为什么还会这么火。

第一简单性，JS 是一门非常轻量级的语言，适合任何人学习，没有大量复杂的保留字，没有各种复杂的数据类型。难怪有人 [探讨]JavaScript 是性价比最高的技术?

第二速度，Google 的 V8 JS 引擎让开发者可以在客户端和服务端更好的执行 JS 代码，这就给复杂的 JS 程序提供了基础。

第三与服务器交互较少，JS 是运行在客户端的脚本语言，这就节省了 Web 服务器的请求时间。另外，可以再提交页面到服务前对用户输入的内容进行验证。这样减少了服务器的通信量，就意味着节约了金钱。

第四跨平台，JavaScript 是依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可以正确执行。

第五让 Web 界面更丰富，尤其是一些带 UI 的 JavaScript 框架，不仅简化 Web 操作，还可以构建非常漂亮的用户界面，例如 ExtJS，其提供了非常丰富的 UI 组件，包括高性能的数据表格、图表、选项卡、弹窗、工具条和菜单等等，可以帮助你构建用户体验良好的 Web 应用。

除了这些优点外，JS 还非常地灵活，作为 JavaScript 程序员，只要你愿意，可以把程序写得很简单，当然，也可以写得很复杂。此外，它还支持多种不同的编程风格。你既可以采用函数式编程风格，也可以采用更复杂一点的面向对象编程风格。■本文未完，完整部分请浏览

http://developer.51cto.com/art/201301/375489.htm

如果编程语言是一条船的话

如果把编程语言想象成一条船的话，它会是那种呢？

1.java

java 是一条货轮，它非常庞大，非常的工业化，虽然他能将装载整个项目，但是驾驭它不是那么简单的。

2.perl

perl 它是一条拖船，有时候能替代货轮。

3.ruby

ruby 很难驾驭，但是很时髦、性感、开起来也很爽。

4.php

PHP 是条竹筏，黑客可以在其中来回穿梭，就像绳子把竹片来回绑紧一样，惊讶的是仍然屹立不倒

5.c

C 语言就是个核潜艇，虽然操作指南可能是外国文字，但船本身性能太棒了，没得说。有点像中国买俄罗斯核潜艇的感觉。

6.html

HTML 压根就不是船。■

编者按

回眸 2012，越来越多的能人志士走向了创业的道路。今天我们就为大家盘点 2012 年的 X 个失败创业案例。希望大家从中能够得到一些启发，对 2013 年新确立一个新的目标有所帮助。

2012年那些失败的创业项目

回眸 2012，越来越多的能人志士走向了创业的道路。现在这个时代，创业是一个热词，实现梦想、创造产品、融资、上市、暴富等诱惑让很多人都走上了创业这条道路。但是创业之路并不都是一帆风顺，这条路上并不是所有的人都能成功。

能记住的只不过是凤毛麟角，大多数的创业项目都在沉寂中期待突破。正如鲁迅说得那样，要么在沉默中爆发，要么在沉默中灭亡。据 17Startup 统计，截止 12 月 23 日 24 点，共收录 6491 家公司，已关停的网站数量达到 643 个，接近十分之一。在这些创业公司中，既有曾风光一时的公司，也有自诞生以来就无人问津的项目，既有当下热门的电子商务、社交网络、云计算，也有已经烂大街的却看不到出路的类型。

其实，吸取失败创业公司的原因要比听成功的经验更受用，那些活生生的案例更让人警醒。也许你在创业的过程中，已经发生这样的事情或者有出现这种状态的苗头。也许有些公司失败的原因我们已经听了很多次，但是它们还是接二连三的发生了。

今天我们就为大家盘点 2012 年的 X 个失败创业案例。

希望大家从中能够得到一些启发，对 2013 年新确立一个新的目标有所帮助。

失败原因 :盲目夸张，缺乏管理

典型案例 :团宝网

团宝网于 2010 年 3 月上线，团购业务率先开通 368 座城市，高峰期，总员工达 2300 人，每天提供超过 2 万个团购选择，拥有超过 1800 万会员，月独立 IP 数亿，日独立 IP 过千万，在百度风云榜和 Alexa 用户访问量排名上位列榜首。曾经是国内最具规模的团购网站，也是国内唯一一家实现 2010 年度盈利的团购网站。我们还记得在北京的写字楼里，到处是团宝网的宣传广告，三位代言人频频出现在各种地方。在当时没有人怀疑团宝网会出现大问题。在当时百团大战如火如荼，各家团购网站都纷纷聘请代言人，大肆做广告，以进一步扩张知名度。

然而，在 2012 年春节前夕，却传出了团宝网 CEO 任春雷跑路的消息。团宝网全国 30 家分站绝大多数均遭裁撤，400 多位员工离职但尚未在公司承诺期限（1 月 20 日）内拿到工资和补偿；数千商家尾款被恶意拖欠；400 服务电话已无人

2012 年那些失败的创业项目 II

接听；北京总部已是变卖殆尽，人去楼空；任春雷及其妻何晓玲手机关机，不知踪迹。

不过任春雷还是回来了，他强调自己并没有跑路，而是去想办法了。春节期间，他和多位投资人见面，均以失败告终，据薛蛮子透露，由于任春雷巨亏，其大股东已经不信任他了。还好他算是一个负责的创业者，尽管公司已经十分困难，还是想办法找投资，还清欠款。并积极寻求团宝网转型。

失败分析：

动辄上亿美元的融资让整个团购行业陷入到迷失当中，在巨额数字面前已经忘记了如何合理利用它们。也许当时团购的火热和投资人的狂热遮掩了这一行业即将出现的危机。创业公司的投资来得太快，让它们忘记了公司需要核算成本。“他们把一个智力游戏简化成了体力劳动，创业过程从超常规模的融资、招聘、广告毫无过渡地走向再融资失败、裁员、倒闭。在集聚了人类杰出智慧的互联网上，他们表现得像一群腰兽皮的史前狩猎者。”当前钱烧完之后，整个行业突然进入冬天，投资方拒绝继续投入。资金链的断裂让公司陷入危机。其中一个投资人表示，他看不出 3 个月后死与现在死的区别，而现在死还能省下一笔钱。

团宝网的盲目扩张导致自身的失败。”有分析认为，包括团宝在内，团购网站几乎都犯了同样的错误。在 .com 时代，互联网创业有三段论：规模做到第一，就可以最快上市，上市之后就可以解决一切问题。但是，这三段论并不适合现在的互联网，更不适合团购。团购行业的净利率很低，竞争激烈时，甚至是赔本赚吆喝，自身的造血能力很低。如果没有资本的支持，那么死亡的概率很大。团宝就犯了这样的错误。虽然这个错误整个团购都犯了，但是由于团宝没有找到第三轮融资，因此，这样的错误对团宝就是致命一击。

另外，也有业内人士认为，团宝在管理上的混乱、不规范尤为突出。在向外人吆喝的过程中，团宝网并没有练好内功，是“自己淘汰了自己”。当创业公司员工超过 2000 时，管理层缺乏管理经验。在快速扩张的时期，每个管理人员都需要花费大量的时间在人员招聘，带新人上。由于精力的大量消耗，许多人在业务上，以及内部管理上有些力不从心。但是，在人员大扩张的时候，正是需要加强管理的时候，而各业务部门的总监经常在外地四外奔波，任春雷四处寻求融资。这让整个公司的中层管理经常处于“真空”的状态。同时，中层、高层人员的离职也让管理团队不稳定，新招来的员工也对团宝没有认同感，忠诚度不高，加之团队成员非常年轻，缺乏一种责任感。

在公司发展上，团宝也在寻求团购差异化。但从结果来看，这些创新收效甚微，有些甚至还是“搬起石头砸自己的脚”。2011 年 9 月 28 日，团宝网推出效仿 Swoopo 竞拍模式的“财迷老道”网站，但是这一活动不仅不赚钱，甚至连用户的眼球都没有赚着。针对流量下滑，任春雷曾经做过一次变革，将团购项目 3 天结束，改为长期在线，永不结束。这让团购项目太多，每个项目能够在首页上展示的机会大大减少了。并且，那些好的项目却无法得到充分的展现，这让用户的体验大幅下降，营业额也大幅下滑。

2012 年那些失败的创业项目 III

现状：

据任春雷透露，新团宝已于 12 月 5 日凌晨 0:00 上线，此前的域名 groupon.cn 已经自动跳转到新域名 tuanbao.com。新团宝网将转型为开放的团购服务平台，主要为用户提供项目发布、展示、购买、营销多种服务，支持商家和消费者自主发布团购信息和项目，任介绍，“新团宝继承了消费者和商家的交易形式，做成了开放平台，可以让更多的人自由出入，以消除行业弊端。”而在企业、代运营者获得收益后，团宝网获得 10% 的服务费，这也是团宝网收入的主要来源。

对于团宝提出的“淘宝式”团购平台模式，业内人士反应不一，其中不乏不同的声音，更有人质疑新团宝是否会昙花一现。“这种模式没新意，做的有点像反向团购，和阿里巴巴所提出的电商未来走向 C2B 类似。……目前只能说是空中楼阁。市场环境还没达到，另外平台不够强大，供应链、物流、支付、信息流、服务管控、售后都没建成完善的体系。……为什么要选择团宝网，而不是聚划算呢？”

失败原因 :在同质化的大潮中淹没

典型案例 :闪聚

去年 8 月，互联网著名人士刘兴亮宣称以“拿起一块名叫闪聚的板砖，狠狠地敲开了移动互联网的大门”，高调宣布其移动互联网创业项目“闪聚”，同年 11 月，iOS 公测版上线，其为一款 LBS 移动交友应用，通过共同感兴趣的约会帮助近距离陌生人结识相聚，主打陌生人交友。刘兴亮此前曾表示创业资金来源于一位老友，足够维持一年的运营开支，后被传融资约 500 万元人民币。

然而在一年过后，据内部人士透露，“闪聚”早在数月前关闭，其在苹果 App Store 中国商店也已经下架，尚存于 Android 商店的 App 也被爆有闪退、无法注册等多处严重 BUG。据知情人士透露，今年 5 月底，闪聚发布 iPhone 1.2 版本后，便再无消息出处，不久后团队宣告解散，创业历程不足 1 年。闪聚官网 (shanju.com) 已经无法打开，联系电话显示为空号，官方微博最新一条消息停留在今年 5 月 21 日。

失败分析：

作为知名的互联网人士，刘兴亮不乏广泛的人脉；作为公司高管，他不乏管理经验；作为互联网评论员，他不乏对业界的认识和发展趋势。但是尽管有这么多的优势来创业，却依旧避免不了创业失败。而闪聚的失败原因却是我们经常谈论的一个原因 :同质化。

2011 年，LBS 移动交友的确是一个非常热门的项目，接二连三地出现了众多产品。这些产品不乏大公司的大制作，也不乏创业公司开发的小产品。但是它们面临着两个巨大的隐患，腾讯和同质化。微信的崛起以迅雷不及掩耳之■

原文未完，请参考

http://developer.51cto.com/art/201212/374237.htm

如何利用一个周末时间成为前端工程师的

我认为一个前端工程师需要的唯一东西是一个好用的 API，和一个全面的文档。作为一个后台开发者，你必须提供这两个。所以，前端的工程师的水平取决于你。但是，不要迫使你的前端人员去使用你的工具。

2 周前，我将 TravisLight 开源了，这是一个建立监控的工具，也是一个用于 Travis-CI 的构件监控工具。出于兴趣，我用了一个周末的时间做了这个项目，而且我是以一个真正的前端开发者的角色来做的。

当我在 Nelmio 的时候，尽管我做了一些 javascript 的应用，但我参与的更多是后台开发。大部分时间我是在写 API 给前端调用，而且在工作中经常会出现偏差。实际上，我认为每一个做后台的开发者应该花些时间在前段开发上，补充一些前端知识，反之亦然。这是让你明白在工作中自己真正需要做些什么的最好的方法之一。

我认为一个前端工程师需要的唯一东西是一个好用的 API，和一个全面的文档。作为一个后台开发者，你必须提供这两个。所以，前端的工程师的水平取决于你。但是，不要迫使你的前端人员去使用你的工具。的确，Assetic 对于 PHP 是一个很好的开发工具，但是它对于前端工程师来说并不是一个好的工具。有许多更好的开发工具可以编译 JS/CSS 文件，写 JavaScript，例如 Grunt。让你前端工程师使用自己的工具吧！在一个 Symfony2 项目中，我会命令把所有的 JS/CSS 文件都放在 web/ 的文件夹中，而不是放在 *Bundle/Resouces/public 这样的文件夹中，这样，前端就不用为了找 JS/CSS 文件夹而去浏览整个项目了。

但这不是我写这篇文章的目的，让我来解释我为什么和怎样写出 TravisLight 和我发现的工具吧。

最初

我想学习一下 Backbone.js，因此我着手阅读了 Backbone Fundamentals 这本书。如果你对 Backbone 还不甚了解，那我给你介绍下，Backbone 是一个 JavaScript 框架，它可以为你编写 Web 程序提供一个架构 .

众所周知，在项目中实践是最佳的编程学习方式，因此，我决定使用 Travis-CI API 去写一个 Backbone.js 应用程序，也就是 TravisLight 。TravisLight 正是那个我一直想要的，用来管理我的开源项目的简明工具。这真是一个绝佳的起步项目，尤其适合在周末进行。

我使用了 Lo-Dash，一个 Underscore.js 的替代品，它具有风格一致，定制灵活，性能优越等优点。同时，我也使用了 RequireJS 和 Moment.js。这样，我就需要一个工具去管理所有的这些依赖，因此我关注了下来自 Twitter 的 Bower。

Bower, 网页包管理器

Bower 是一个网页包管理器，也就是 JS/CSS 库包管理器。虽说它现在是一个包下载器，但还是有必要用它来避免对 jQuery, Twitter Bootstrap

如何利用一个周末时间成为前端工程师的 II

等进行版本控制。你只需要一个像这样的 component.json 文件：

```
{
  "name": "travis-light",
  "dependencies": {
    "jquery": "~1.8.3"
  }
}
```

运行“bower install”把依赖关系安装到组件或文件夹。现在，我可以开始着重弄我的应用。每次我需要一个新的库只要运行“bower install--save”来安装和更新“component.json”文件就行了。

有时候，我需要使用一些工具帮助我在应用上完成一些像运行 jshint 或编译文件的任务。于是我试了“Grunt”——一个 Javascript 编译工具，感觉还不错哦。

Grunt, JavaScript 编译工具

Grunt 是一个基于任务的命令行 JavaScript 工程编译工具 . 第一眼看上去 , 这个工具似乎难以使用，但是一旦你用了，它太棒了！你能够验证 (Lint) 你的文件，缩小 JS/CSS 文件，运行测试单元等等。

在 TravisLight，我主要使用 Grunt 打包应用程序 . 打包应用程序意味着 :

编译 JavaScript 文件 ;

编译 CSS 文件 ;

在 HTML 标记中使用编译好的文件 ;

复制依赖库 .

编译 Javascript 文件就是编译 RequireJS 的依赖关系。幸亏有 grunt-contrib-requirejs 插件，太简单了！

```
requirejs: {
    compile: {
        options: {
            name: "main",
            baseUrl: "js/",
            mainConfigFile: "js/main.js",
            out: "dist/compiled.js"
        }
    }
}
```

在 TravisLight 里面编译 CSS 有两步：

首先、把 CSS 里面的所有图片用 grunt-image-embed 插件嵌进来：

```
imageEmbed: {
    application: {
        src: 'css/application.css',
        dest: 'dist/application-embed.css',
        deleteAfterEncoding : false
    }
}
```

然后，用 grunt-contrib-mincss 插件压缩 CSS 文件；

```
mincss: {
    compress: {
        files: {
            'dist/compiled.css': [
                'css/bootstrap.min.css',
                'dist/application-embed.css'
            ]
        }
    }
}
```

现在，为了使用那些编译过的 JS 和 CSS 文件，我用 grunt-targethtml 插件编译 HTML。

```
targethtml: {
    dist: {
        src: 'index.html',
        dest: 'dist/index.html'
    }
}
```

■原文未完，请参考：

http://developer.51cto.com/art/201212/374902.htm

编者按

在刚刚过去的 2012 年中，Java 的形象因为安全方面的问题大受影响，但它仍然是一个关键的企业软件开发平台。而 2013 年，用户可以期待新的版本发布，包括企业版 Java 和标准版 Java。

经过糟糕一年后，Java正沿着正确方向发展

在刚刚过去的 2012 年中，Java 的形象因为安全方面的问题大受影响，但它仍然是一个关键的企业软件开发平台。而 2013 年，用户可以期待新的版本发布，包括企业版 Java 和标准版 Java。

JDK 8 基于 Java 平台标准版 8，将提供一些新特性包括 JavaScript 编程和对多核处理器的支持。同时 Java EE 7 将更加易用以及实现新的 HTML5 WebSocket 通讯支持和 RESTful Web 服务 2.0 规范实现。

这两个即将到来的版本中包含的功能是毋庸置疑的，JavaScript 已经成为事实上的 Web 开发标准，而现在已经是多核计算机的时代了。HTML5 和 RESTful Web 服务业在软件开发和 Web 服务领域日渐流行。

与此同时，用户需要升级到 Java SE 7 版本，相应的 Java SE 6 将在 2 月份停止支持，这个已经推迟两次了。

2013 年以后，Java 将在 Java SE 9 中更加模块化，更加适合云计算环境。

虽然 Objective-C 和很多动态语言获得了不少的关注，但 Java 仍是最流行的编程语言，包括 PYPL 排名中 Java 仍是最受欢迎的语言，而 TIOBE 排名中 Java 排在第二。在 Dice 技术和工程专业人员网站中预计 2013 年 Java 开发职位的需求仍是最多的，这已是连续第二年排在第一位置。

所有的信息显示，将满 18 岁的 Java 编程语言仍是与时俱进的。

热点专题

根据W3C的发言稿称：“HTML5是开放的Web网络平台的奠基石。”
这种跨平台的网页程序环境通常被称之为“Web标准”的保护伞。

链接：http://developer.51cto.com/art/201212/373250.htm

编者按

陈述的技巧有很多，坦率的说我是非常不擅长技巧的人，所以我就给出一个原则吧：更早的提醒风险，总是比到事情最后发展到无法收拾再处理要好。

张中：工程师进阶之路

工程师进阶之路(一)

做一个专业而亲切的人。

最近这本书给了我启示《12 Essential Skill for Software Architects》。那就慢慢写下读书笔记吧。

那么怎样才能做到泥?

保持关系，而不是一味的纠正别人

工程师由于每天都在和大量的程序、机器打交道，工作的重心之一就是订正各种各样的错误，但是人不是软件更不是机器，一味的去订正别人的错误是非常有害的(而且我们当时认为的“错误”其实更多情况下是我们没有全面审视而得出的结论，随着社会压力、生活节奏加快，这点尤其突出)。在纠正别人之前，先学会问自己两个问题?

1. 这个“错误”是不是非常重要?

2. 暂时忽略这个错误会不会给公司或者组织带来重大影响?

如果答案是“NO！”，那么我们应该停止：打断别人去纠正”错误“的行为。

学会授权和委派

这一点我深有感受，不仅对于一线技术管理者来说，就是对资深的老工程师来说或多或少都有问题。面对问题，工程师天然的反应就是”我自己能否解决?“”我如何解决?“，但是正确的授权和委派不仅仅能够让项目进展更顺利，更关键的是能够建立信任，让不同层次的人都能得到成长。

生活是一面镜子

你的言行是正向、友善、积极的，那么你收到的反馈就是正向、友善、积极的。你的言行是负向、带刺、消极的，那么你就会觉得全世界都在为难你。工程师要走好他的技术道路一定要意识到这一点，这也是为什么许多工程师“跳槽”或者“换岗”之后，还是不适应，很大的一个原因就是他没有意识到改变自身行为模式的重要性。

注意语言的表达

有的时候我会收到别的同事的反馈说，为什么你们有的工程师这么“梗”，经过了解和交谈，我发现原来是很多工程师不好的口头禅造成的，比如“这个不行啊！”，“你连这个都不懂啊”，“我没有时间！（然后就没有反馈了）”。语言就像种子一样，你播种下去就会长起来的，你播种的是带刺的藩篱，那么它长大后就会把你困在其中。这个和阿里巴巴价值观里面常说的“直言有讳”是一个道理。

及时处理出现的问题

这个更多的是指“软”问题，呵呵借用了一下“软技能”。当我们工程师每天面临各种纷繁复杂的技术问题在处理的时候，不自觉的就忽略了这种软问题的处理，比如我是不是遇到了一个很“难缠”的合作伙伴，我是不是在一个多方合作的项目中感到无助和困惑，我是不是和上司间存在

张中 :工程师进阶之路 II

认识的不统一和误解，我是不是在该说“No！”的时候选择了沉默？千万别埋葬这些问题，因为很快它又会出现，直到无法收拾。我是一个内向的人，沟通不是我的强项，但是不意味着放弃沟通；我是一个“Nice”的人，但是不意味着从来不说“No！”。及时处理出现的问题，意味着我们在不停的挑战自己，在修行中成长。

提供专业的服务

怎么理解专业的服务呢？不仅仅是专业的IT技能，就像我们判断一个酒店是否专业一样，不仅是看大堂还要看它的客房，还要看它的接待，还要看它的氛围。专业是全方位的。因此我们工程师除了要会用代码服务的时候，还要学会：

微笑

◆合适的手势表达

◆把大家都加入到谈话中来

◆关注别人，而不是以自我为中心

◆别一心二用，身在曹营心在汉

◆随时准备给别人提供帮助

◆学会关心他人

◆学会倾听、

原谅并忘记曾经的冒犯

这个不多说，还记得我们曾经也冒失过吗？还记得曾经也小人之心过吗？那么我们遇到的冒犯都是浮云啦！呵呵。

工程师进阶之路 二

谈谈沟通能力——沟通的准则

如果一名工程师要成长为资深专家或者是架构师或者是技术管理者，沟通是必不可少的技能和工作的工具。

对于资深专家或者架构师，对沟通能力的要求甚至大于技术管理者，因为他们没有行政权力，但是却往往需要主导、指导、控制跨小组、跨团队乃至跨公司的项目，怎么能够把各个组织里面的人有效的驱动起来，沟通能力非常重要。

但是我们工程师在成长过程中，往往不注意或者没有收到这方面的训练，当他技而优则”升“成为专家的时候，问题就来了，尤其是他作为团队中坚，既要和一线工程师交流又要向老板汇报，真的是压力山大啊！

沟通方式和技巧往往是因人而异的，但是我们还是可以总结出一些基本准则供大家参考。

先说一下沟通的准则：

先听后说

古罗马哲学家 Epictetus 曾经说过很有意思的一句话，人之所以要少说多听，是因为我们只有一个嘴巴但有两个耳朵。

我们古人也总结了：兼听则明偏信则暗，进一步补充了耳朵各在左右两边的重要性，哈哈。

马云对还没有成为三年阿里人说的话里面，第一句就是多看少说。

当然他们更多的是从宏观的角度来论证，其实走入到具体的沟通微观例子中也是同样适用的。

我曾经一开会就立马紧张起来，为什么啊？是因为要急于表达自己的观点和成绩，深怕别人不知道，总是想见缝插针的说话，结果是别■

本文 未完，详细请查看：

http://developer.51cto.com/art/201301/375657.htm

王远轩：北美求职记

作者 / 王远轩

最近签掉了 offer，找工作的事情算是告一段落。在这里写一点面试体验和心得，希望对有兴趣去北美工作的朋友有所帮助。

先简单介绍下自己，国内硕士在读，明年毕业，没有牛 paper，也没参加过 ACM-ICPC 竞赛。在实验室做过内核、虚拟机和 Android 底层相关的研究工作，接过一些网页和移动开发的外包，2011 年开始在字节社兼职负责后台开发。另外也经常上 Stackoverflow 和 GitHub。

这次决定直接申请美国的职位后，由于心里没底，不知道国外公司招聘的难度，所以一开始投了很多公司。几个大公司都找人内推或者直接投了，小公司也投了不少，比如 Foursquare、Path、Pinterest 和 Square 等都试了。当时甚至在手机上找了一圈应用，把可能涉及后端开发的应用都投了一遍。不过大多数公司都没给我安排面试，只有 Microsoft、Google、Facebook、Twitter 和 Hulu 这五家公司愿意给我面试机会。

一般来说，国内毕业后直接投国外公司，会比出国留学毕业后找工作的难度大一些。除了语言因素之外，我了解到的主要原因在于工作签证，出国留学毕业后可以通过 OPT 签证入职，之后再过渡到 H-1B 签证。而国内毕业的学生只能通过 H-1B，这意味着要等到第二年的十月份才能入职。好在 Google、Facebook 等公司不太介意这个问题，还是会欢迎国内的应届生申请。

校招的 HR 一般会有各自的职责。比如 technical sourcer 负责发现有希望进入自己公司的应届生；recruiter coordinator 会帮助 recruiter 安排面试者的面试时间、面试官，以及 onsite 面试时帮助面试者订机票和酒店；staffing consultant 则负责发 offer 以及介绍公司的具体福利制度，并解释面试者相关的问题。不同公司的 HR 职责的分法自然也不一样，我在 Facebook 的面试过程中只和两位 HR 联系过，而在微软的面试过程中则联系过五六位 HR。

在面试流程方面，相比我了解到的国内公司的面试，国外公司的面试安排上会更人性化一些。例如安排面试时间时，HR 一般会先让你给出几个空闲的时间点，然后他们再从这些时间中给你安排面试。此外在为你安排 onsite 的住宿时，也会询问你有没有相关的要求。

关于面试题目，大多数公司都比较侧重面试者对基本的数据结构和算法的掌握程度，以及把这些内容实现为实际代码的能力（一般会要求你选一个语言实现，而不允许用伪代码）。越是规模大的公司越注重这些基本功，而小公司除此之外还会考察你的开发经验，例如对某个框架的了解和性能优化方面的技巧。关于这一点区别我的理解是大公司里面会有自己的框架和开发工具，面试者的基本功好就能比较快的上手；而小公司一般用社区现有的工具，所以已有的开发经验可以直接用在将来的工作中。

下面是这几个公司的面试细节，有些公司因为在 onsite 面试的时候签了 NDA，所以没法透露具体的面试题，还请见谅。

王远轩 :北美求职记 II

Microsoft

微软是我最早投的公司之一,托了在微软总部工作的一位学长帮忙内推。面试包括一轮 HR 面和四轮 onsite 面。

申请了一个多月后一直都没有反应,直到微软国内招聘的前一天,北京的 HR 打电话问我是不是投过微软的职位,要我参加第二天上海站的笔试。

笔试过后,又过了一个多月,收到了微软一位招聘人员的邮件,问我是不是对微软北美的职位有兴趣,要我填一份基本情况的问卷,里面有问到其他公司的面试进度。我当时已经收到了 Google 和 Facebook 的面试邀请,就如实填写了。回复第二天后就收到了邮件通知,告诉我会有 HR 进一步跟进。第三天有一位 HR 联系我和我约电面的时间。微软约电面的方式和其他公司不大一样,HR 会给出很多个选项,让你在里面选择几个空闲的时间。另外值得一提的是这些时间都转成北京时间了,这也是微软在安排面试时比较人性化的一个地方。

第一轮面试是 HR 面。HR 先问了一些技术无关的问题,比如喜欢做什么,工作地点的偏好,什么时候开始学的编程,为什么投了微软等等。接着是一些智力题,比如 9 个小球,8 个质量相等,另一个比其他的重,如何用天平称两次把它找出来 ;公司开发了一种新键盘,有哪些测试它的方法 ;在会议室内怎么估计室外的温度。都是些更像是考验英语水平而不是技术能力的问题。

面完第二天收到了 onsite 的通知。虽然是北美的职位, onsite 面试地点却是在上海。我参加的是周日的面试,和我一起参加面试的还有一位学生,他之前在微软实习,了解到这次有去北美工作的机会后也想尝试下。面试官是从总部飞过来的工程师,一共有四位,其中三位都已经是 principal 级的了。HR 提到一般技术面试要五轮,因为我们之前参加过一轮笔试,所以只需要面四轮。

onsite 面每一轮的过程都差不多,都是面试官自我介绍,接着我介绍自己和做过的一些项目,然后开始技术问题,最后是我提问的环节。微软的面试问题会考察面试者编码、设计和测试三方面的能力。

coding 环节要求直接在白板上写代码,我被问到两个 coding 问题。一是如何检查一棵二叉搜索树是否正确,二是写一个解数独的程序。第一个问题写起来很快,第二个问题因为时间有限,我先写了一个没啥剪枝的暴力搜索的版本,写完后和面试官分析了可以在此之上做的优化。

设计方面的问题有两个。第一个问题是设计一个分布式的数据管理系统。使用场景可以是一个连锁店信息的记录系统,每个分店都有可能更新自己的信息,并把这些改动传播到整个系统中。在设计这一系统的同时要考虑性能、容错、一致性等要求。

我一开始想了一个基于 push 的机制,在面试官指点下逐步优化,最后还是有不少问题。于是干脆重新设计了一个基于 poll 的系统,优化改进之后面试官满意了。

另一个设计问题和类的设计有关,要求设计一个包含图形界面的棋盘游戏。

■本文未完,详细请查看 :
http://developer.51cto.com/art/201212/375354.htm

如果没有末日：2013年十大热点技术发展趋势

作者 / 王帅

编者：如果没有末日，2013 年科技车轮将继续滚滚向前，新一年会有哪些科技？将继续引领业界潮流？看看由赛迪顾问股份有限公司 王帅为我们独家整理的 2013 年十大热点技术趋势。

---

1. 3D 打印

从平民化的 3D 电影到工业革命 2.0 的 3D 打印机，3D 技术已经普及到个体并正在进一步发展。2012 年是 3D 电影全面普及的一年，如果说 3D 电影带给人们视觉的冲击，那么 3D 打印机带给人们想象力的刷新。吉他、牛排、枪支、机械臂、房子、飞机这些都已经是 3D 打印机的成果。3D 打印技术将是美国振兴制造业的有力武器。可以估计的将来，3D 打印机将“变出”我们想要的物品，帮助我们在火星建立基地、甚至打印染色体和克隆人。

2013 年是 3D 打印技术进一步发展的一年，将呈现三大趋势：

一 . 3D 打印设计软件展现易用性。

二 . 3D 打印材料标准将受到重视。

三 . 打印成果将朝着大而精发展。

2. HTML5

HTML5 作为第五代超文本标记语言，包含了 HTML 4,XHTML 1 和 DOM Level 2 HTML，将成为互联网领域的新一代生力军。在电子设备移动化的趋势下，它是对付 iOS 和 Android 系统 APP 的有利武器，不仅更好的支持多媒体播放和拥有 API，而且基于 HTML5 开发的 APP 将不受系统平台限制。根据 W3C（万维网联盟）的计划，2013 年是 HTML 5.0 标准接受审阅的一年，尽管发展缓慢，HTML5、CSS3 和 JavaScript 的组合将高端化 HTML 应用并成为 Flash 强有力的对手。

2013 年是 HTML5 技术完善发展的一年，将呈现三大趋势：

一 . 开发和发布更快捷和容易。

二 . 网页和应用速度得到优化。

三 . 更丰富、互交的用户体验。

3. 物联网

无论是 M2M（机器对机器）通信应用，还是 NFC（进距离通信）技术，都是物联网的组成部分。这种基于传感器的网络有别于因特网，如果说因特网是信息的网络，那么物联网更强调物与物的联系。尽管信息流是物联网关键的一部分，但是物联网的直接信息来源多数是机器而非人类，物联网中的每个物体将成为微型主机，自动感知并收集信息，有的处理后再被发送到需要的地方，这个地方往往是另一个电子设备。智能家居、智能电网、智能物流等等都是物联网的具体应用。

2013 年是物联网技术广泛渗透的一年，将呈现三大趋势：

如果没有末日 :2013 年十大热点技术发展趋势 II

一 . IPv6 技术在物联网中崭露头角。

二 . 多种无线通信技术遍布终端应用。

三 . 物联网中的物体拥有地理位置信息。

4. 大数据

大数据是 Google 超大服务器群储存的搜索信息,大数据是社交网站用户发表的图片、文字和视频等,大数据也是历年各地气候状况及农作物产量。大数据的快速发展离不开因特网和物联网,正是网络上日益猛增的数据让大数据技术水到渠成。大数据技术的目标是利用数据进行快速准确的业务分析、业务决策、业务发展以及业务拓展。对于大数据,获取是前提,分析是核心,决策是目的。大数据分析就是将零散片面的数据变成统一深刻的有价值的信息的过程。

2013 年是大数据技术深度应用的一年,将呈现三大趋势 :

一 . 获取数据手段自动化和智能化。

二 . 实时分析过程依赖内存分析。

三 . 决策越来越多是数据和算法驱动。

5. 云计算

云计算是一种集中计算资源、按需分配的网络托管技术。云端拥有各类硬件和软件,它既可以是公有云也可以是私有云,十分便利的硬件拓展性和软件延伸性使云计算广受欢迎。云计算是网络中的中枢神经,负责获取数据后及时的处理和稳定的反馈。大到操作系统,小到网盘,都能在云中找到应用。大数据技术侧重的是 IT 系统中的数据层,云计算技术侧重的是 IT 系统中的应用层,SDN 技术侧重的是 IT 系统中的网络层。

2013 年是云计算技术迅速发展的一年,将呈现三大趋势 :

一 . 安全性逐渐得到大众的认可。

二 . 实时分析过程依赖内存分析。

三 . 移动设备涌现为云计算终端。

6. SDN

以前的网络安全、协议、路由、能耗管理都被封装在设备中,很难进行大规模的改动。随着大数据和云计算的发展,数据中心和分布式系统对网络的易用性和拓展性要求越来越高,SDN(软件定义的网络)是 IT 系统中网络部分的创新,它将网络的逻辑层(控制层)与表现层(网络拓扑及数据)分离,可以有效的解决网络的冗余和延伸,而且能够方便的管理 VLAN(虚拟局域网)和 VM(虚拟主机)。SDN 弹性的解放在硬件配置上的注意力 , 使得更为重要的网络服务被凸现出来。

2013 年是 SDN 技术初步应用的一年,将呈现三大趋势 :

一 . 基于 SDN 的应用迈向专业化。

二 . SDN 和云计算技术相互渗透。

三 . 网络设备朝着少而精发展。

7. 4G

4G 作为移动电话系统第四代,由一系列的标准和技术组成,如 WiMAX、LTE。它与 3G 最大的区别在于 4G 是基于 IP 协议的和由此收获的高数据传输速度,但是 4G 通信是兼容 3G、2G 通信的。如今,平板电脑和智能手机的普及使得 PC 行业每况愈下,人们对移动网络的要求也■

本文未完,详细请查看 :

http://developer.51cto.com/art/201212/373673.htm

分辨软硬需求？90后创业经历反思

到底什么是刚性需求？为什么有时候觉得这是强需求，而用户却觉得这些可有可无？为什么很多需求看起来是“强需求”“硬需求”，做起来后却发现是“弱需求”“软需求”？这是一个让很多创业者头疼的问题，在这我分享一下自己之前两次创业的经历，希望能引起一些共鸣。

本文作者为南京洛哈网络科技 CEO（微博），他正在创业路上。

到底什么是刚性需求？为什么有时候觉得这是强需求，而用户却觉得这些可有可无？为什么很多需求看起来是“强需求”“硬需求”，做起来后却发现是“弱需求”“软需求”？

这是一个让很多创业者头疼的问题，在这我分享一下自己之前两次创业的经历，希望能引起一些共鸣。

一，软需求：一语“唤醒”梦中人

大一的时候，我和几个同学开始了一次创业，产品是“校园清洁”，我们联系好第三方清洁公司，然后在网上搭建一个平台，最后在校园里做地推宣传，帮大学生去清洁宿舍。

这个简单的项目，在美国大学做的话，是一个不错的机会。为什么这么说呢？

1. 美国大学宿舍是没有清洁大婶的，纯靠学生自己整理

2. 美国大学的宿舍一般是“公寓式”（四室一厅一卫一厨）或“别墅式”，公寓里的厨房和卫生间很难打扫，加之美国人懒惰的习性。而别墅式更不必说了，打扫一次至少花去一天时间

3. 美国大学生爱喝酒爱 party，每周末屋子里总是满地酒瓶、厨房里铺满了杯子盘子和 pizza，更别忘了卫生间的呕吐物……

根据我们前期的市场调查，判断出美国学生应该急需这种一周一次或两周一次的宿舍清洁服务：因为考虑到安全把关、信息透明等问题，学生很难在校外找到靠谱的服务（偷走了什么 iPhone 和 Xbox 可就坏了），而如果由我们作为中间方，提供的服务保障肯定靠谱多了。而波士顿又是著名的“大学城”，五步一 College、十步一 University，市场容量极大、消费水平也很高，如果能建立一套成熟的体系，前途不可限量（至少当时这么认为的……）

想到就做，很快我们产品就运营了。刚开始时生意十分红火，我们对所有宿舍挨个地宣传（特别是周日早晨，当那些宿舍的主人在满地垃圾中醒来的时候，最需要的就是一位勤劳的墨西哥清洁大婶了），同时收费也不算高，平摊下来每个人只要付 5-10 美元。

当一切都向着好的方向去发展时，情况却产生了变化——越来越多的学生取消了服务。发生了什么？我们的服务不好？安全有问题？还是收

分辨软硬需求？ 90 后创业经历反思 II

费太高？一问发现都不是，而他们的答案很简单“因为你们的服务，我们意识到了自己的宿舍很脏，但同时我们也发现打扫其实不难。为什么不能从现在起自己打扫呢？”——之前学生对宿舍清洁确有需求，但当有产品满足这个需求时并收费时，消费者立刻意识到他们其实并没有必要为这个需求买单。随后的结果不言而喻，这个我们认为的“硬需求”成了泡影，同学们都开始自觉清理宿舍了。

我们想满足用户的需求，最后却教育了用户、提醒了用户，用户反而去寻找更简单的方法解决这类需求，这是大多“软需求”的共性。

二，硬需求：一语“吓醒”梦中人

大二我又一次创业。有了之前几次创业的经历，对团队运营更加熟练了一些，对产品、市场和需求也有了更好的理解，这次选取的市场依旧是波士顿的大学生，产品模式依旧是做线上线下的“中间平台”（也就是现在流行的 O2O）。

这次是做日用品的“物流配送”，主要考虑到以下几点：

1. 学校物价出奇的高，一瓶水卖 2.99 美元，而 Costco、沃尔玛里的 36 瓶水特价时也只卖 2.99 美元！物价相差数十倍，这是一个很强的差价市场；

2. 大学生很多人没车，就不可能出去购物，而美国的公共交通又不方便，并且一个人很难扛回来许多东西；

3. 大学生的日用品消耗量很大，与此同时，大学生很懒。

统计了需求后，开发、上线、租车、采购、配送等环节并不困难，很快产品就进入市场了，学生们只要上网站预订即可，从矿泉水到香水、从卫生纸到避孕套，应有尽有。一开始的时候，订单寥寥无几，由于日用品的重要性，一般学生不敢托大。

而随后我们设计出了一系列的推广口号，如“订购一个‘月包装（Monthly Package）’只要 99.99 美元，比学校便宜了 450%！”、“你每月在采购生活用品上平均花费了 14.75 个小时，这够你和她约会 5 次了！”，此类的宣传标语渐渐让学生意识到这是一个切实存在、却无法解决的需求！后来，越来越多的学生使用了此项服务，并且二次使用率达到了接近 100% 的程度。

两次创业，相同的市场、相似的产品、相近的需求，却截然不同的结果。虽然经验和团队方面有一定影响，但我个人感觉两次创业“成也需求、败也需求”，前者需求很“软”而且可替代性极强，而后者对用户产生了“恐慌式需求”，让他们不得不用这个产品、用了以后还得再用。

生活中这样的例子比比皆是，比如说携程，酒店价格排序的巨大差异和靠谱程度的两极分化，让用户有着极强的依赖性——很多用户一看性价比上有这么大的差距，敢不用吗？就跟没有点评不敢吃饭一样，少了 OTA 网站很多人绝对不敢住店了。而格瓦拉也是一样，之前去排队买票是天经地义的事，但是一有网上订票以后大家会发现“再去排队这是得有多蠢啊”——况且别人在网上订而你不订的话，那你肯定就只能坐角落了。而说到反面例子，其实“餐厅订座”这个东西一直在做，包括小秘书和饭统网以及一些地方网站，但是目前都很难做大，我猜可能主要的■

本文未完，详细请查看：

http://developer.51cto.com/art/201212/373679.htm

HBase性能优化的四个要点

作者 / 石头儿

1 hbase.hregion.max.filesize 应该设置多少合适？默认值 :256M

说明 :Maximum HStoreFile size. If any one of a column families' HStoreFiles has grown to exceed this value, the hosting HRegion is split in two.

HStoreFile 的最大值。如果任何一个 Column Family（或者说 HStore）的 HStoreFiles 的大小超过这个值，那么，其所属的 HRegion 就会 Split 成两个。

调优：

hbase 中 hfile 的默认最大值 (hbase.hregion.max.filesize) 是 256MB，而 google 的 bigtable 论文中对 tablet 的最大值也推荐为 100-200MB，这个大小有什么秘密呢？

众所周知 hbase 中数据一开始会写入 memstore，当 memstore 满 64MB 以后，会 flush 到 disk 上而成为 storefile。当 storefile 数量超过 3 时，会启动 compaction 过程将它们合并为一个 storefile。这个过程中会删除一些 timestamp 过期的数据，比如 update 的数据。而当合并后的 storefile 大小大于 hfile 默认最大值时，会触发 split 动作，将它切分成两个 region。

lz 进行了持续 insert 压力测试，并设置了不同的 hbase.hregion.max.filesize，根据结果得到如下结论 :值越小，平均吞吐量越大，但吞吐量越不稳定 ;值越大，平均吞吐量越小，吞吐量不稳定的时间相对更小。

为什么会这样呢？推论如下：

a 当 hbase.hregion.max.filesize 比较小时，触发 split 的机率更大，而 split 的时候会将 region offline，因此在 split 结束的时间前，访问该 region 的请求将被 block 住，客户端自我 block 的时间默认为 1s。当大量的 region 同时发生 split 时，系统的整体访问服务将大受影响。因此容易出现吞吐量及响应时间的不稳定现象

b 当 hbase.hregion.max.filesize 比较大时，单个 region 中触发 split 的机率较小，大量 region 同时触发 split 的机率也较小，因此吞吐量较之小 hfile 尺寸更加稳定些。但是由于长期得不到 split，因此同一个 region 内发生多次 compaction 的机会增加了。compaction 的原理是将原有数据读一遍并重写一遍到 hdfs 上，然后再删除原有数据。无疑这种行为会降低以 io 为瓶颈的系统的速度，因此平均吞吐量会受到一些影响而下降。

综合以上两种情况，hbase.hregion.max.filesize 不宜过大或过小，256MB 或许是一个更理想的经验参数。对于离线型的应用，调整为 128MB 会更加合适一些，而在线应用除非对 split 机制进行改造，否则不应该低于 256MB。

2 autoflush=false 的影响

无论是官方还是很多 blog 都提倡为了提高 hbase 的写入速度而在应用代码中设置 autoflush=false，然后 lz 认为在在线应用中应该谨慎进行该设置。原因如下：

a autoflush=false 的原理是当客户端提交 delete 或 put 请求时，将该请求在客户端缓存，

HBase 性能优化的四个要点 II

直到数据超过 2M(hbase.client.write.buffer 决定) 或用户执行了 hbase.flushcommits() 时才向 regionserver 提交请求。因此即使 htable.put() 执行返回成功,也并非说明请求真的成功了。假如还没有达到该缓存而 client 崩溃,该部分数据将由于未发送到 regionserver 而丢失。这对于零容忍的在线服务是不可接受的。

b autoflush=true 虽然会让写入速度下降 2-3 倍,但是对于很多在线应用来说这都是必须打开的,也正是 hbase 为什么让它默认值为 true 的原因。当该值为 true 时,每次请求都会发往 regionserver, 而 regionserver 接收到请求后第一件事就是写 hlog,因此对 io 的要求是非常高的,为了提高 hbase 的写入速度,应该尽可能高地提高 io 吞吐量,比如增加磁盘、使用 raid 卡、减少 replication 因子数等

3 从性能的角度谈 table 中 family 和 qualifier 的设置

对于传统关系型数据库中的一张 table,在业务转换到 hbase 上建模时,从性能的角度应该如何设置 family 和 qualifier 呢?

最极端的,①每一列都设置成一个 family,②一个表仅有一个 family,所有列都是其中的一个 qualifier,那么有什么区别呢?

从读的方面考虑:

family 越多,那么获取每一个 cell 数据的优势越明显,因为 io 和网络都减少了。

如果只有一个 family,那么每一次读都会读取当前 rowkey 的所有数据,网络和 io 上会有一些损失。

当然如果要获取的是固定的几列数据,那么把这几列写到一个 family 中比分别设置 family 要更好,因为只需一次请求就能拿回所有数据。

从写的角度考虑:

首先,内存方面来说,对于一个 Region,会为每一个表的每一个 Family 分配一个 Store,而每一个 Store,都会分配一个 MemStore,所以更多的 family 会消耗更多的内存。

其次,从 flush 和 compaction 方面说,目前版本的 hbase,在 flush 和 compaction 都是以 region 为单位的,也就是说当一个 family 达到 flush 条件时,该 region 的所有 family 所属的 memstore 都会 flush 一次,即使 memstore 中只有很少的数据也会触发 flush 而生成小文件。这样就增加了 compaction 发生的机率,而 compaction 也是以 region 为单位的,这样就很容易发生 compaction 风暴从而降低系统的整体吞吐量。

第三,从 split 方面考虑,由于 hfile 是以 family 为单位的,因此对于多个 family 来说,数据被分散到了更多的 hfile 中,减小了 split 发生的机率。这是把双刃剑。更少的 split 会导致该 region 的体积比较大,由于 balance 是以 region 的数目而不是大小为单位来进行的,因此可能会导致 balance 失效。

而从好的方面来说,更少的 split 会让系统提供更加稳定的在线服务。而坏处我们可以通过在请求的低谷时间进行人工的 split 和 balance 来避免掉。■

本文未完,更多内容请查看原文:

http://database.51cto.com/art/201301/376723.htm

航旅纵横-基于民航的数据整合之路

在 2012 年 12 月 4 日 Velocity 大会上，51CTO 专访了来自航空行业的唐老师。唐老师在本次大会上给大家带来的是跟海量民航异构信息处理的技术分享。

航旅纵横技术负责人唐先生本次分享中谈到了航旅纵横海量异构数据整合的问题，这些异构数据的来源有国内外民航系统的差异、也有部分是各地机场 IT 系统水平参差不齐，即使民航多年来 IT 系统的不断提升，但人员技能方面的差异、基础网络的稳定性等还是会造成异构数据问题。在这个问题上，民航等传统行业与信息行业有很大的不同。信息行业的技术革新一般都是颠覆性的，而传统行业大多都是打补丁的形式，必须对原有技术系统进行继承。这一点上，会对整体信息系统的效率造成一定的影响。

航旅纵横为旅客提供便捷的航空服务，其在数据整合过程中，会有一个数据信息取舍的问题。如何整理一份及时准确的数据，需要对源数据的字段进行有效识别，某些字段需要保留，某些字段必须舍弃。在这个平衡度的问题上，团队采用的方法是根据历史数据和流程，确定需要保留的字段。这是一种目前在用的思路。

不断增加的数据量级

中国民航业在不断发展，航线信息和旅客量在成倍增加。这些都会成为一个海量数据的来源，并且随着航空服务形式的多样性和社交媒体的爆发，航空数据不可避免的加入越来越多非结构化的数据。未来民航业将会是一个大数据集中的行业，该如何来处理这些数据呢？

首先数据涉及很多敏感数据，需要进行高层次的保护措施以免泄露，保护信息的安全。同时对数据进行分析和挖掘，为旅客和企业提供相应的信息服务。对于数据的融合，采用系统隔离，规则的筛选和清洗，数据出口的数据校验等方式，形成了较为自动的数据模型。

对于 IT 技术人员，在设计系统架构时，还应考虑极端情况（比如恶劣天气等）系统的影响。唐老师表示自己的团队会考虑在现有系统的处理能力基础上，做出 2 倍到 3 倍的性能冗余。如果这些冗余都无法保证正常的运行时，会保证绝大部分用户的正常使用，而会把一些不重要的用户流量进行暂停、可以保证系统的安全和正常运行。

同时还谈到了廉价航空的发展，让更多的旅客可以享受更加廉价的机票。而对于廉价航空公司，未来的民航服务将会实现模块化。根据其需要进行选择，从而减小整体的 IT 服务成本。

移动设备是提升航空服务的利器

罗永浩与国航空乘的对峙事件在微博上吵得沸沸扬扬。其实较大程度是沟通不畅和信息不同步造成的原因，在今天移动互联高速发展的时代，国内航空企业已经在逐步开始推广移动设备终端吗，未来各种定制化的信息会更加精确的送到航空公司空乘人员的手中，让他们更好的为旅客服务，较大的提升用户的出行幸福指数。■

MariaDB成为MySQL命运转折点?

当初Sun Microsystems公司即将迎来收购之时,一群曾经参与过著名人气开源数据库MySQL开发的程序员们决定另起炉灶,打造名为MariaDB的新项目。

新项目由Michael“Monty”Widenius定名并领导,这位MySQL项目的原始开发者兼MySQL公司联合创始人放弃甲骨文的招揽,从零开始重新奋斗。在离开Sun公司之后,他在自己的故乡荷兰成立了一家公司——也就是Monty Program AB——借以管理MariaDB项目的开发工作,同时向广大MySQL技术达人敞开怀抱。不久之后,Monty Progam公司就拥有了一个实力强劲的开发团队。

也许大家并不了解,但他们的确一直在废寝忘食地工作。由于甲骨文公司在全面收购Sun资产后对MySQL的开发工作表现出极高热情,导致MariaDB感受到了前所未有的竞争压力。然而优秀的人才在对抗当中力挽狂澜,帮助MariaDB站稳了脚跟。在一份由Network World网站公布的六大开源数据库评测报告中(包括MySQL),我们发现MariaDB赫然成为人气最高的数据库方案。Monty告诉我,MariaDB与MySQL相比拥有约三十人工作年的研发优势,而Monty Program公司也对自家产品的领先性表示认同——尤其是在安全性修复方面。

快速修正是关键

Monty告诉我们,MariaDB开发团队一直在与mitre.org通力合作,希望保障一切安全问题都能被快速发展、上报且拥有完备的细节描述。由于甲骨文公司不再公布安全修复细节,MariaDB团队通常需要对来自MySQL的补丁程序进行逆向工程,借以找出其修复对象。搞清状况之后,这些补丁将被合并起来并为MariaDB提供服务。Monty表示“MariaDB可以被看作是安全性最高的MySQL版本”——这一声明相当大胆。

尽管MariaDB与MySQL两者在新版本公布之前,其安全漏洞的修复细节一般都要受到严格保密,但最近的一次事件令双方在安全应对机制与响应速度上的差异显露无遗。问题甫一曝光,MariaDB就行动起来并在几天之内就利用开放并记录开源补丁完成了修复工作;相比之下,MySQL直接现在(截稿之日)仍然没能搞定这些安全漏洞。

这已经不算什么新鲜事了。甲骨文公司一直在对企业资源优先参与并处理MySQL社区事务的做法抱怨不已,结果当然显而易见:裁撤相关流程转投其它开发项目并延缓修复安全漏洞,而这一切都令项目组与MySQL生态系统间的交流日益恶化。我曾与MySQL与MariaDB双方的外部生态系统合作伙伴进行过对话,而他们都对甲骨文公司的强硬作风表示无奈。

面临如此窘境,上周传出的新闻无疑可算大大的利好消息:MariaDB基金会正式成立,MySQL开源社区也将有望自此步入新的发展阶段。MariaDB——从侧面来看也就是MySQL——终于拥有了专门的机构体系,该基金会的出现将

MariaDB 成为 MySQL 命运转折点？ II

一举扭转发展控制一家掌握的被动现状。

生态系统受到影响

这对 MySQL 的生态系统又意味着什么？首先，MySQL 将借打包与整合之力获得提升。很明显，像 MariaDB 这样的开放式项目在使用便捷性上要远超过某家企业针对自身业务所打造的数据库方案。在未来，我们很可能看到被囊括在 Linux 发行版中的 MariaDB，同样也可能在 LAMP 部署中发现它的身影（而且在用 MariaDB 代替传统 MySQL 之后，我们仍然可以沿用‘M’这个字母）。

其次，创新的可能性也将大大增加。某位开发人士曾告诉我，MariaDB 身上具备一些极具吸引力的发展潜力——支持 OLTP、OLAP、以数据为中心的专业处理方案以及高度可扩展的多控制集群。这种多元化的发展方向意味着项目需要接纳来自各个方面的信息与意见，而基金会的成立为开发者提供了一套透明的管理平台，其向所有人开放的基础特性对于整个项目的走向有着非常深远的积极意义。

第三，有效改善项目竞争力。MariaDB 最近刚刚公布了一套兼容性极高的客户端库，完全利用 LGPL 从零开始重新编写、借以替代 MySQL 及其衍生版本原先所使用的 GPL——现在 MariaDB 与 MySQL 双方都将由此而受益。正如社区成员 Arjen Lenz 在评论中所说，这一点对于双重许可问题意义重大。现在商业用户们再也不必为了避免额外的 GPL 合规性管理需求而忍痛为 MySQL 购买专有许可了。

将三个因素综合起来看，MariaDB 绝对有机会在规模庞大且对手众多的 MySQL 市场中依靠独特魅力取得竞争优势。Monty 告诉我们，MariaDB 项目的代码贡献者中已经出现了许多企业巨头的身影，其中包括 Facebook、谷歌、Twitter 等，而且专为开发者们准备的 IRC 交流平台上经常有上百人在线。

虽然形势见好，但也并非万事大吉。MySQL 生态系统中有那么一部分似乎就完全没受到正面影响 :Drizzle 项目，其目的是通过重新设计一套更小、更具模块化特性的微内核实现云部署。该项目创始人 Brian Aker 在 Twitter 上明确表达了自己对于基金会的不屑一顾，并通过电子邮件宣称自己将保持观望态度。Drizzle 项目已经拥有了自己的一套非 GPL 客户端库——虽然还未能实现完全兼容——而 Aker 认为 MariaDB 所使用的 JDBC 驱动（即 Java 数据库连接）从某种程度上来说源自 Drizzle（其采用 BSD 许可，而 MariaDB 则采用 LGPL）。

在任何大型技术社区当中，意见分歧都是不可避免的，所以我们倒是无需对这种剑拔弩张的气氛太过在意。

相信只要拥有良好的执行力，MariaDB 基金会很可能为 MySQL 社区注入新的活动，并实现又一轮创新与业务增长。要达到这一目标并不简单，但项目中所蕴含的潜力同样极为可观。MySQL 已经成为开源历史上的一块重要基石，而 MariaDB 很可能在未来的发展道路上成为又一座丰碑。■

分布式文件系统HDFS设计

HDFS 有几个关键性的词组 :Very large files, Streaming data access, 以及 Commodity hardware。解下来一个一个解释。

1 Very large files

在 Hadoop 中, “very large” 是多大? 运行在 HDFS 上的应用具有很大的数据集。HDFS 上的一个典型文件大小一般都在 G 字节至 T 字节。现如今,已经有不少的企业,存储在 HDFS 上的数据,已经超过了 PB 的级别,例如淘宝。

2 Streaming data access

HDFS 的设计的理想用法是一次写入,多次读取。这种方式的使用是最高效的。运行在 HDFS 上的应用和普通的应用不同,需要流式访问它们的数据集。HDFS 的设计中更多的考虑到了数据批处理,而不是用户交互处理。比之数据访问的低延迟问题,更关键的在于数据访问的高吞吐量,所以,从这方面来说,读取整个数据的时间延迟要比读取到第一条记录的数据延迟更重要(这一点很重要,这一思想,将在《HDFS 学习(二) HDFS Block 介绍》中说明)。

3 Commodity hardware

Hadoop 适合部署在廉价的机器上,即普通的机器上,不需要时昂贵且高可靠的机器。这样的话,机器节点的故障几率就会非常高。所以, Hadoop 是一个高度容错的系统,错误检测和快速、自动的恢复是 HDFS 最核心的架构目标。Hadoop 出现故障时,被设计成能够继续进行且不让用户察觉。

综上, HDFS 是一个不错的分布式文件系统,但是, HDFS 也有其不适合的场合,也有其缺点 :

1 低延时数据访问

HDFS 不太适合于要求低延时(数十毫秒)访问的应用程序,因为 HDFS 是设计用于高吞吐量数据访问的,这就需要以一定的延时为代价。而对于那些有低延时要求的应用程序, HBase 是一个更好的选择。HBase 的口号就是 “Use Apache HBase when you need random, realtime read/write access to your Big Data”。还有一个问题就是,因为 Map task 的数量是由 splits 来决定的,所以用 MR 处理大量的小文件时,就会产生过多的 Map task,线程管理开销将会增加作业时间。处理大量小文件的速度远远小于处理同等大小的大文件的速度。举个例子,处理 10000M 的文件,若每个 split 为 1M,那就会有 10000 个 Map tasks,会有很大的线程开销 ;若每个 split 为 100M,则只有 100 个 Map tasks,每个 Map task 将会有更多的事情做,而线程的管理开销也将减小很多。

对于第一问题,最新版本的 Hadoop 已经有了解决方案 :HDFS Federation,将在《HDFS 学习(四) - HDFS Federation》做详细介绍。

对于第二个问题, Hadoop 本身也提供了一定的解决方案。

3 多用户写,任意文件修改

目前 Hadoop 只支持单用户写,不支持并发多用户写。可以使用 Append 操作在文件的末尾添加数据,但不支持在文件的任意位置进行修改。《Hadoop 权威指南》第三版上说,现在尚无对这方面的支持。■